2007

Formal Security Evaluation of Ad Hoc Routing Protocols

Todd R. Andel
THE FLORIDA STATE UNIVERSITY
COLLEGE OF ARTS AND SCIENCES

FORMAL SECURITY EVALUATION OF AD HOC ROUTING PROTOCOLS

By

TODD R. ANDEL

A Dissertation submitted to the Department of Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Degree Awarded:
Fall Semester, 2007
The members of the Committee approve the Dissertation of Todd R. Andel defended on November 13, 2007.

____________________________________
Alec Yasinsac  
Professor Directing Dissertation

____________________________________
Michelle Kazmer  
Outside Committee Member

____________________________________
Sudhir Aggarwal  
Committee Member

____________________________________
Breno de Medeiros  
Committee Member

____________________________________
Gary Tyson  
Committee Member

Approved:

____________________________________
David Whalley, Chair  
Department of Computer Science

____________________________________
Joseph Travis, Dean, College of Arts and Sciences

The Office of Graduate Studies has verified and approved the above named committee members.
ACKNOWLEDGEMENTS

First and foremost, this thesis would not have been possible without the gifts and talents given to me by God and the support provided to me from my wife. I thank my wife for accepting the enormous interruption on our family life. There are not enough words of thanks I can provide her for all the encouragement and understanding she has provided me during this program.

Secondly, I thank the faculty and staff, who made this program possible. Special thanks goes to my advisor Dr. Alec Yasinsac for his continual academic, professional, and personal guidance.

Finally, I thank the U.S. Air Force and the Air Force Institute of Technology for providing this fellowship opportunity. The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.
# TABLE OF CONTENTS

**LIST OF TABLES** .................................................................................................................. vi

**LIST OF FIGURES** ........................................................................................................... vii

**ABSTRACT** ........................................................................................................................... ix

**CHAPTER 1 INTRODUCTION** ............................................................................................... 1

1.1 Problem Definition .............................................................................................................. 1

1.2 Research Goals and Objectives ......................................................................................... 3

1.2.1 Adaptive Threat Modeling ............................................................................................ 3

1.2.2 Automated Security Analysis ......................................................................................... 3

1.3 Chapter Summary .................................................................................................................. 4

**CHAPTER 2 AD HOC ROUTE SECURITY** ........................................................................... 5

2.1 Ad Hoc Routing .................................................................................................................. 5

2.1.1 Single-Phased Routing ................................................................................................. 6

2.1.2 Two-Phased Routing .................................................................................................... 7

2.2 Security Issues .................................................................................................................... 8

2.2.1 Secure Routing and Data Forwarding ......................................................................... 8

2.3 Chapter Summary ................................................................................................................ 11

**CHAPTER 3 CURRENT EVALUATION TECHNIQUES** .................................................... 12

3.1 Non-Exhaustive Evaluation Approaches .......................................................................... 13

3.1.1 Visual Inspection ........................................................................................................... 13

3.1.2 Network Simulation ...................................................................................................... 15

3.2 Exhaustive Evaluation Approaches .................................................................................... 17

3.2.1 Analytical Proof ............................................................................................................ 17

3.2.2 Simulatability Models .................................................................................................. 18

3.2.3 Formal Methods .......................................................................................................... 21

3.3 Chapter Summary ................................................................................................................ 27

**CHAPTER 4 BOUNDING THE EVALUATION** ............................................................... 29

4.1 Towards a Comprehensive Security Evaluation Framework ........................................ 29

4.2 Focusing on Route Discovery ............................................................................................ 31

4.3 Isolating Malicious Failures .............................................................................................. 32

4.4 Chapter Summary ................................................................................................................ 33

**CHAPTER 5 ADAPTIVE THREAT MODELING** ................................................................ 35

5.1 Attacking Route Discovery ............................................................................................... 36

5.1.1 Attack Sources .............................................................................................................. 36

5.1.2 Classical Routing Attacks ............................................................................................ 36

5.2 Canonical Threat Modeling .............................................................................................. 37

5.2.1 The Classical Dolev-Yao Attacker Model ................................................................... 38

5.2.2 MANET Attacker Models ............................................................................................ 39

5.3 An Adaptive Threat Model ............................................................................................... 41

5.3.1 The Model .................................................................................................................... 42

5.3.2 An Example .................................................................................................................. 45

5.3.3 Classifying Current MANET Attacks ......................................................................... 47

5.4 Chapter Summary ................................................................................................................ 48

**CHAPTER 6 AUTOMATED SECURITY ANALYSIS** ....................................................... 49

6.1 The SPIN Model Checker ................................................................................................ 50

6.2 Modeling the Wireless Medium ....................................................................................... 54

6.3 Model Validation ................................................................................................................ 58

6.4 Modeling Source Routing Protocols ............................................................................... 59

6.4.1 Modeling DSR .............................................................................................................. 59

6.4.2 Modeling SRP .............................................................................................................. 62

6.4.3 Modeling Ariadne ....................................................................................................... 65

6.4.4 Modeling endairA ....................................................................................................... 70
LIST OF TABLES

Table 1. Security Evaluation Approaches ................................................................. 12
Table 2. Formal Methods used in MANET Security Analysis .................................................. 22
Table 3. SAODV Topology Consistency Failures ............................................................. 23
Table 4. Comprehensive Security Analysis Framework ....................................................... 30
Table 5. Attacker Classification ....................................................................................... 42
Table 6. Unique Network Topologies .............................................................................. 81
Table 7. Hardware and Software Analysis Components .................................................. 86
Table 8. Global System Requirements – Worst Case ....................................................... 95
Table 9. SPIN Runtime Requirements ........................................................................... 96
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MANET Communication</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Ad Hoc Routing</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>DSR Route Discovery</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>Routing Protocol Operations</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>Visual Inspection Cycle</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>Cell Grid Overlay</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>Simulatibility Modeling</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>Real vs. Ideal Network Connectivity</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>Calculus Modeling Relationships</td>
<td>22</td>
</tr>
<tr>
<td>10</td>
<td>Topology Inconsistency Attack</td>
<td>23</td>
</tr>
<tr>
<td>11</td>
<td>Petri Nets</td>
<td>24</td>
</tr>
<tr>
<td>12</td>
<td>CPAL-ES Protocol Evaluation</td>
<td>25</td>
</tr>
<tr>
<td>13</td>
<td>SRP CPAL Attack Scenario</td>
<td>25</td>
</tr>
<tr>
<td>14</td>
<td>NSPK and Lowe Attack Strand Representation</td>
<td>26</td>
</tr>
<tr>
<td>15</td>
<td>SAODV Attack Analysis</td>
<td>27</td>
</tr>
<tr>
<td>16</td>
<td>Classical Dolev-Yao Attacker</td>
<td>38</td>
</tr>
<tr>
<td>17</td>
<td>Dolev-Yao MANET Attacker</td>
<td>39</td>
</tr>
<tr>
<td>18</td>
<td>Combining Local Attackers</td>
<td>40</td>
</tr>
<tr>
<td>19</td>
<td>Asymmetrical Attacker</td>
<td>43</td>
</tr>
<tr>
<td>20</td>
<td>Single Attacker Example</td>
<td>45</td>
</tr>
<tr>
<td>21</td>
<td>Example Attack Sequence</td>
<td>46</td>
</tr>
<tr>
<td>22</td>
<td>Automated Security Analysis Process</td>
<td>50</td>
</tr>
<tr>
<td>23</td>
<td>Modeling Message Reception</td>
<td>51</td>
</tr>
<tr>
<td>24</td>
<td>Exhaustive State Space</td>
<td>53</td>
</tr>
<tr>
<td>25</td>
<td>MANET Model Abstraction</td>
<td>55</td>
</tr>
<tr>
<td>26</td>
<td>Representing Network Topology</td>
<td>57</td>
</tr>
<tr>
<td>27</td>
<td>High Level Wireless Medium Server Process</td>
<td>57</td>
</tr>
<tr>
<td>28</td>
<td>High Level DSR Route Discovery Process</td>
<td>60</td>
</tr>
<tr>
<td>29</td>
<td>Path Validation Topology</td>
<td>61</td>
</tr>
<tr>
<td>30</td>
<td>DSR Path Validation</td>
<td>61</td>
</tr>
<tr>
<td>31</td>
<td>SRP Protocol Messages</td>
<td>62</td>
</tr>
<tr>
<td>32</td>
<td>High Level SRP Route Discovery Process</td>
<td>63</td>
</tr>
<tr>
<td>33</td>
<td>SRP Path Validation</td>
<td>64</td>
</tr>
<tr>
<td>34</td>
<td>Five-node Attacker Topology</td>
<td>65</td>
</tr>
<tr>
<td>35</td>
<td>SRP MAC Validation</td>
<td>65</td>
</tr>
<tr>
<td>36</td>
<td>Ariadne Protocol Messages</td>
<td>66</td>
</tr>
<tr>
<td>37</td>
<td>High Level Ariadne Route Discovery Process</td>
<td>67</td>
</tr>
<tr>
<td>38</td>
<td>Ariadne Path Validation</td>
<td>69</td>
</tr>
<tr>
<td>39</td>
<td>Ariadne Hash and Signature Validation</td>
<td>70</td>
</tr>
<tr>
<td>40</td>
<td>endairA Protocol Messages</td>
<td>71</td>
</tr>
<tr>
<td>41</td>
<td>High Level endairA Route Discovery Process</td>
<td>71</td>
</tr>
<tr>
<td>42</td>
<td>endairA Path Validation</td>
<td>72</td>
</tr>
<tr>
<td>43</td>
<td>endairA Signature Validations</td>
<td>73</td>
</tr>
<tr>
<td>44</td>
<td>High Level SRP NDA Process</td>
<td>76</td>
</tr>
<tr>
<td>45</td>
<td>High Level Ariadne NDA Process</td>
<td>77</td>
</tr>
<tr>
<td>46</td>
<td>Modeling Attacker Storage Space</td>
<td>78</td>
</tr>
<tr>
<td>47</td>
<td>High Level endairA NDA Process</td>
<td>79</td>
</tr>
<tr>
<td>48</td>
<td>Network Connectivity Array</td>
<td>80</td>
</tr>
<tr>
<td>49</td>
<td>5-Node Symmetric Topology Array Values</td>
<td>82</td>
</tr>
<tr>
<td>50</td>
<td>Binary String Representation</td>
<td>82</td>
</tr>
<tr>
<td>51</td>
<td>Topology Generation Process</td>
<td>83</td>
</tr>
</tbody>
</table>
ABSTRACT

Research into routing protocol development for mobile ad hoc networks has been a significant undertaking since the late 1990’s. Secure routing protocols for mobile ad hoc networks provide the necessary functionality for proper network operation. If the underlying routing protocol cannot be trusted to follow the protocol operations, additional trust layers cannot be obtained. For instance, authentication between nodes is meaningless without a trusted underlying route.

Security analysis procedures to formally evaluate these developing protocols have been significantly lagging, resulting in unstructured security analysis approaches and numerous secure ad hoc routing protocols that can easily be broken. Evaluation techniques to analyze security properties in ad hoc routing protocols generally rely on manual, non-exhaustive approaches. Non-exhaustive analysis techniques may conclude a protocol is secure, while in reality the protocol may contain unapparent or subtle flaws. Using formalized exhaustive evaluation techniques to analyze security properties increases protocol confidence.

Intertwined to the security evaluation process is the threat model chosen to form the analysis. Threat models drive analysis capabilities, affecting how we evaluate trust. Current attacker threat models limit the results obtained during protocol security analysis over ad hoc routing protocols. Developing a proper threat model to evaluate security properties in mobile ad hoc routing protocols presents a significant challenge. If the attacker strength is too weak, we miss vital security flaws. If the attacker strength is too strong, we cannot identify the minimum required attacker capabilities needed to break the routing protocol.

To solve these problems, we contribute to the field in the following ways:

**Adaptive Threat Modeling.** We develop an adaptive threat model to evaluate route discovery attacks against ad hoc routing protocols. Adaptive threat modeling enables us to evaluate trust in the ad hoc routing process and allows us to identify minimum requirements an attacker needs to break a given routing protocol.

**Automated Security Evaluation.** We develop an automated evaluation process to analyze security properties in the route discovery phase for on-demand source routing protocols. Using the automated security evaluation process, we are able to produce and analyze all topologies for a given network size. The individual network topologies are fed into the SPIN model checker to exhaustively evaluate protocol models against an attacker attempting to corrupt the route discovery process.

Our contributions provide the first automated exhaustive analysis approach to evaluate ad hoc on-demand source routing protocols.
CHAPTER 1
INTRODUCTION

Mobile ad hoc networks (MANETs) are an attractive solution for wireless networking, eliminating the reliance on fixed networking infrastructure. Unlike wired networks that depend on fixed routers for network connectivity and message forwarding, each wireless MANET node provides routing functionality. Ad hoc networks provide an alternative for military applications and disaster response scenarios where fixed networking infrastructure may not be available.

Ad hoc routing protocols provide the essential functionality for MANET operation, allowing a node to communicate with the nodes that are beyond its local transmission distance. As secure routing protocols are being developed, they must be analyzed to ensure the protocol’s intended security requirements are maintained. Unfortunately, most MANET research in the literature generally follows unstructured approaches or non-exhaustive methods to evaluate route security. These approaches tend to consistently result in a stamp of approval from the protocol developer, as they claim to show their protocol has met their defined security criteria. The effect is a false sense of security, since many of these secure protocols are later shown to have flaws.

The contributions contained in this thesis provide the first automated exhaustive analysis approach to evaluate ad hoc on-demand source routing protocols. The thesis aims to advance the understanding of MANET routing protocol vulnerabilities prior to mainstream protocol deployment and use. Our primary research contribution provides an automated evaluation process to analyze security properties in the route discovery phase for on-demand source routing protocols. We follow the formal methods approach by providing an exhaustive automated model checking technique to evaluate ad hoc route discovery against active route corruption. We additionally develop an automated topology generation and analysis process to examine all possible network topologies for a given network size \( N \) and develop an adaptive threat model to ensure analysis results are not limited to restricted adversarial capabilities. These contributions complement the current non-exhaustive and non-automated MANET security analysis approaches to provide a more comprehensive security analysis capability to evaluate security properties in MANET routing protocols.

1.1 Problem Definition

MANETs face significantly more security considerations than wired networks due to their dynamic nature and inability to confine a wireless transmission signal to a specified user population. Mobile wireless communication is highly susceptible to malicious activity, since
broadcast wireless communication can be eavesdropped and valid nodes may be physically captured and compromised.

In the late 1990's and early 2000's research into MANET routing protocols [1] primarily focused on functionality and efficient operation. Once an afterthought, protocol security is now being addressed, as many proposed secure ad hoc routing protocols are being researched [2,3,4].

Routing protocol security is vital for proper MANET operation in malicious environments. We consider a protocol secure if it is accurate and reliable in the face of attackers. In order for a MANET routing protocol to operate properly, we must trust intermediate nodes making up a routing path to follow protocol rules. Trusting intermediate nodes to follow protocol rules is critical in MANET implementations since these networks are highly dynamic. Nodes may continuously join or leave the network and network connectivity changes with mobility. Additionally, wireless message transmission is highly vulnerable to malicious activity. Broadcast wireless communication can be eavesdropped and nodes may be physically captured and comprised. If a malicious node can corrupt or inject itself into the routing path, message deliverability depends on the attacker's intentions.

Analyzing MANET security properties presents a considerable challenge. Proving unconditional protocol security is impossible, since it is generally considered infeasible to exhaustively evaluate a protocol to determine if it's vulnerable against unknown attacks or future attacker capabilities. Security analysis must focus on how a protocol's intended security goals guard against the threat environment in which the protocol is intended to operate. Unfortunately, security analysis within current MANET literature predominately follows unstructured or non-exhaustive evaluation techniques. In their survey on secure ad hoc routing protocols, Hu and Perrig [2] draw attention to the fact that the secure routing protocol problem is not properly modeled. They pose the idea that a more formalized attacker model would assist protocol security evaluation and allow formal methods development to analyze MANET protocol security. Argyroudis and O'Mahony [3] reinforce the call to formally model MANET routing protocols to provide a solid mathematical security definition, allowing researchers the ability to formally prove that a protocol does or does not satisfy its intended security goals.

Additional obstacles to MANET security analysis arise by using overly restrictive or idealistic assumptions and current research lacks common security definitions. While all evaluation approaches require assumptions to form the analysis problem, results based on imprecise assumptions may be misleading. An ad hoc protocol touted as provably secure in one setting may be vulnerable if the assumptions do not hold in the operating environment. The idea of provable security does not imply a protocol is unconditionally secure. This problem is analogous to the software engineering approach to testing in that it is impossible to apply testing to
determine if software is defect free. Quoting Dijkstra [5], "Program testing can be used to show the presence of bugs, but never to show their absence!"

To combat these problems, a comprehensive security analysis capability is required to analyze security properties in MANET routing protocols. A comprehensive approach provides a more thorough understanding into what environments a given protocol is secure and to what vulnerabilities we know a protocol cannot protect against.

1.2 Research Goals and Objectives

To solve the problems facing MANET security evaluation, our thesis is that a comprehensive security analysis capability to evaluate security properties in MANET routing protocols will provide a significant advancement in understanding protocol vulnerabilities.

To meet the thesis goal we pose the following research objectives:
1. Develop an adaptive threat model.
2. Develop an automated security analysis technique.

1.2.1 Adaptive Threat Modeling

Modeling the MANET routing protocol threat environment presents a considerable challenge that demands applying assumptions on the attacker capabilities. The attacker strength may range from the attacker having the same power and capability of a non-malicious node to assuming the attacker has no communication or computational limitations as viewed in the classical Dolev-Yao [6] attacker model.

The canonical approach to MANET security evaluations uses fixed attacker capabilities, limiting the results to the specified environment. If the attacker capabilities are too weak, we miss critical security flaws. If the attacker capabilities are too strong, we cannot determine the minimum required capabilities required to break the protocol.

In addition to modeling attacker capabilities, the MANET threat model must also consider the security goals for the evaluation, with the protocol security goals being tied closely to the intended operating environment.

An adaptive threat model provides the ability to search for security flaws without limiting analysis to restricted attacker capabilities. Adaptive threat modeling allows us to identify minimum requirements an attacker needs to break a given routing protocol.

1.2.2 Automated Security Analysis

Mainstream evaluation approaches in the mobile ad hoc network routing community do not provide an automated or exhaustive security analysis capability. Security evaluation techniques to

---

1 With the exception of breaking cryptographic primitives in polynomial time.
analyze ad hoc routing protocols generally rely on manual or manually assisted approaches. Non-exhaustive analysis techniques may conclude a protocol is secure, while in reality the protocol may contain unapparent or subtle flaws.

A formalized automated and exhaustive evaluation approach can increase protocol confidence. Combing our automated exhaustive process with current security evaluation approaches provides a more comprehensive security evaluation capability.

1.3 Chapter Summary

In this chapter we introduced the problems facing security analysis techniques to evaluate MANET routing protocols. We additionally presented the research goals and objectives to provide a formalized approach to provide an exhaustive automated evaluation capability to analyze MANET route security. The remainder of the thesis is outlined as follows. Chapter 2 provides the required background on ad hoc routing and presents critical security issues facing the MANET routing community. Chapter 3 provides foundational survey material to fully understand current security evaluation techniques used to analyze MANET routing protocols and discusses the limitations faced by the associated approaches. Chapter 4 provides the thesis foundation and scope. In Chapter 5, we present an adaptive threat modeling approach. In Chapter 6 we present development models to provide an automated security evaluation analysis capability. A demonstration of the automated security evaluation process is contained in Chapter 7. Chapter 8 concludes with a summary and discusses future research opportunities.
CHAPTER 2
AD HOC ROUTE SECURITY

This chapter contains required background necessary to frame the thesis research objectives, based on our previous work [7]. Section 2.1 provides background on ad hoc routing, a key element enabling MANET operation. In Section 2.2, we discuss MANET security properties required for proper operation.

2.1 Ad Hoc Routing

Routing is the process of finding a valid route between two nodes and subsequently utilizing the route for data communication. MANET routing faces considerably more challenges than fixed wired networks. Challenges include securing broadcast wireless communication in an un-trusted environment and merely finding the route itself. Routing is further challenged in mobile ad hoc networks. Network topology is not fixed and network membership may rapidly change. Node movement creates a continually changing network topology that MANET routing protocols must consider. Additionally, nodes rely on their neighbors for network topology information and to relay data to nodes outside their local 1-hop transmission distance. Figure 1 illustrates the reliance on intermediate nodes to provide wireless MANET communication between the two gray nodes.

![Figure 1. MANET Communication](image)

Ad hoc routing protocols are frequently classified according to their routing strategy as reactive\(^2\) or proactive\(^3\) [1,2,3]. We take a different approach to classifying routing protocols, as depicted in Figure 2. We classify routing protocols as single-phased or two-phased, based on whether a route must be established prior to transmitting data. Single-phased approaches embed data into the routing process, while the two-phased approach establishes a path before sending data. The single and two-phased classifications have distinctive security requirements, which are discussed in the following two sections.

\(^{2}\) Reactive protocols are sometimes referred to as source-initiated or on-demand.

\(^{3}\) Proactive protocols are sometimes referred to as table-driven.
2.1.1 Single-Phased Routing

Single-phased routing approaches embed data into the routing process. Data is sent within the routing messages without any prior knowledge of routes between a source and its desired destination. As an illustration, assume that data messages are transmitted to their intended destination via flooding.

Flooding is a simplistic approach in which all nodes except the destination rebroadcast every distinct packet they receive [8]. Incoming packets that have previously been received and retransmitted are simply dropped. In non-malicious environments, flooding guarantees deliverability to all connected nodes. Connectivity relies on the network density, node position, and wireless transmission distance. In malicious environments, deliverability is guaranteed if at least one adversary-free path exists. The disadvantage to flooding is communication overhead, since all non-target nodes relay every packet at least once. In dense networks, the increased overhead and wireless channel contention produced in flooding systems can lead to broadcast storms [9]. Flooding is commonly used as the underlying process during the route discovery process in two-phased routing protocols, but can be used to provide single-phased routing if data is embedded into the flooding messages.

To improve efficiency in classical flooding based systems, gossip protocols [10] and their variants [11,12] optimize effort by reducing the number of nodes that retransmit a given packet. The goal of these flooding adaptations is to achieve the fine line between optimizing overhead by reducing transmissions vs. maintaining full network coverage provided by classical flooding.

Protocols such as Location-Aided Routing (LAR) [13] can also reduce flooding requirements. Location-based routing protocols rely on positional information to make routing decisions. The source and destination position are embedded into the route request. Intermediate nodes forward route discovery packets only if they geographically lie between the previous hop and the final destination. LAR schemes are generally used during route discovery in two-phased approaches, but can be used in single-phased schemes.

Single-phased routing protocols are inefficient and encounter significant overhead, since each individually sent data message must determine its own route to the desired destination. Data
messages sent between the same source-destination pair do not capitalize on the fact that subsequent messages can sent over the same route, as long as the route remains fresh.

2.1.2 Two-Phased Routing

Two-phased routing protocols separate communication into route discovery and data communication to combat the inefficiency and overhead encountered in single-phased routing schemes. Route selection reduces overhead by precisely selecting retransmission nodes. Two-phased routing protocols rely on the underlying single-phased operations for route discovery. For example, flooding is commonly used to deliver route discovery messages until the desired route is established. Upon successful route discovery, data packets are subsequently transmitted along the discovered route during the route lifetime. Using established routes thus reduces traffic overhead, as only the nodes identified by the route retransmit data packets.

We use the Dynamic Source Routing (DSR) protocol [14] and Figure 3 to illustrate two-phased routing protocols. Prior to data transmission, DSR performs route discovery to identify a path between a source and destination. The route discovery phase is separated into route request (\textit{rreq}) and route reply (\textit{rrep}) processes. Intermediate nodes process and retransmit a given \textit{rreq} only the first time the node receives the associated request. Once the destination receives the \textit{rreq}, it returns a \textit{unicast} \textit{rrep} along the discovered path, assuming the links are symmetric or bidirectional. To simplify Figure 3, node \(d\) responds to a single \textit{rreq}. However, the original DSR protocol does not specify if only one or all route requests must be responded to with a unique \textit{rrep}.

![Figure 3. DSR Route Discovery](image)

We further refine the two-phased routing classification to specify \textit{reactive} or \textit{proactive} routing strategies [1,3]. \textit{Reactive} routing protocols generate routes as needed. Reactive source-initiated routes may use embedded paths as in DSR, or utilize tables to determine the next hop along the path. For example, the Ad Hoc On-Demand Distance Vector (AODV) protocol [15] is a reactive source-initiated routing protocol that uses tables to store routing information.

---

\(^4\) \textit{Unicast} messages are addressed to individual nodes, as opposed to broadcast messages intended for all nodes.
Proactive routing protocols use tables to identify the next hop required to reach the destination [1,3]. What distinguishes proactive from reactive protocols is that each node in a proactive approach strives to maintain continuous routing information to communicate with all network nodes, as opposed to reactive protocols that construct routes only when needed. Proactive routing protocols may not on the surface appear to follow the two-phased routing process. However, table generation and update mechanisms constitute the route discovery phase. Proactive MANET routing protocol examples include the Destination-Sequenced Distance Vector (DSDV) [16] routing protocol, the Optimized Link State Routing (OLSR) [17] protocol, and the Wireless Routing Protocol (WRP) [18].

Two-phased routing protocols reduce traffic overhead for long messages broken into multiple packets or messages intended for previously discovered destinations, since only the nodes identified in the route retransmit packets during the data communication phase. Unfortunately, if any node along the two-phased route fails, the entire route may fail and must then be reestablished.

2.2 Security Issues

MANET routing protocols are vulnerable to attacks, such as denial of service, packet delay, packet modification, packet dropping, and spoofing. Both the ad hoc routing process and the data communication, or data forwarding, phases must be secured in order to provide a complete solution. MANET secure routing developments within current literature generally target securing either the route discovery phase or data communication phase [4,19] individually. Secure routing protocols must also consider the underlying cryptographic mechanisms to deliver secure solutions.

2.2.1 Secure Routing and Data Forwarding

The 2004 survey by Hu and Perrig [2], the 2005 survey by Argyroudis and O'Mahony [3], and the 2005 survey by Djenouri, Khelladi, and Badache [4] discuss recent literature to develop secure MANET route discovery and data forwarding protocols. Of course authors claim their developed secure routing protocols are in fact secure. However, many of these secure protocols are subsequently shown to be insecure. The discovered vulnerabilities generally result from inappropriate or overly restrictive attacker capability assumptions or simply overlooking a subtle protocol flaw due to incomplete security evaluation techniques.

A contributing factor that complicates MANET secure routing development and analysis is that there is no accepted security definition. Secure routing does not seamlessly map to the canonical confidentiality, integrity, and availability (CIA) [20] security requirements. Any attack that alters MANET protocol operation essentially affects availability. Attacks that result in corrupt routes disrupts route discovery integrity. Maliciously altering routing information is an integrity issue, but
relaying routing messages may not be. Message relay may not violate the traditional integrity
definition, per se, if the relaying node does not alter the message. However, message relay is
malicious if it inhibits the routing protocol from accomplishing its goals.

One common MANET routing security focus is preventing invalid route selection [21]. This
view isolates the attacks against the route discovery phase, but does not consider the effect
malicious insiders may impart on the data forwarding phase.

The other common security view targets securing the data forwarding phase. Attacks against
the data forwarding phase occur when nodes along the discovered route discard or corrupt data
packets. The insider threat is by and large believed impossible to eliminate, since legitimately
operating malicious insiders cannot be detected until they break the protocol rules. Since the
attacker cannot be identified before it acts maliciously, this security view attempts to identify and
avoid malicious nodes during future routing decisions [22]. The goal for security mechanisms for
the data forwarding phase is to converge to a secure system by eliminating insider attackers once
they are identified.

Inconsistent security definitions ensure inconsistent results. Virtually all MANET routing
protocol authors claim their protocol is secure. In reality, their security definitions apply only to
their own boundaries or assumptions. Work in [3] and [4] surveys many proposed MANET
security protocols along with their intended environment and limitations.

Without a common security definition, it is infeasible to compare protocol security properties
and reported attempts are inconsistent throughout the literature. Consequently, we define security
based on whether a protocol meets its intended operational goals. That is, does the protocol
deliver packets in all environments or attacker scenarios? The following three properties must be
maintained for a routing protocol to meet its objectives:

1. **Accuracy.** A routing protocol is accurate if it produces routes that meet its objectives.
   For example, are the returned routes consistent with the current network topology?

2. **Reliability.** A routing protocol is reliable if its returned routes are always accurate,
   even if non-malicious failures occur. For instance, can routes be reestablished if node
   mobility results in link failure?

3. **Security.** A routing protocol provides security if it preserves the protocol's accuracy
   and reliability in the face of malicious attackers.

Figure 4 maps these three properties onto two basic operations for two-phased routing
protocols: a routing protocol must be able to find valid routes by securing the routing process and
must also mitigate route failures once they occur by securing the data forwarding phase.
Secure route discovery. The ad hoc routing protocols discussed Section 2.1.2 assume non-malicious environments. While their goal is to find a route between a source and destination, these protocols do not contain mechanisms to prevent malicious route manipulation. There are numerous proposed secure routing protocols intended to guard against corrupt routes and ensure malicious outsiders are not included in discovered routes. Solutions intended to secure the routing process include the Secure Routing Protocol (SRP) [23], Ariadne [24], Security-Aware ad hoc Routing (SAR) [25], Secure Efficient Ad hoc Distance Vector (SEAD) [26], Secure AODV (SAODV) [27], Authenticated Routing for Ad hoc Networks (ARAN) [28], Secure Position Aided Ad hoc Routing (SPAAR) [29], and the Secure Link State Routing (SLSR) [30] protocols.

Secure data forwarding. Once routes are established via secure route discovery, two-phased routing protocols continue to be vulnerable to attacks against the data forwarding phase. Malicious insiders, or Byzantine [31] attackers, are legitimate routing entities, since malicious insiders are fully trusted and hold certified cryptographic keys. It is impossible to identify a corrupt insider until the node acts maliciously. As long as a malicious node forwards data according to protocol rules, it is not a threat, per se, since the data successfully reaches its destination. Once malicious activity occurs, secure ad hoc routing protocols must take steps to mitigate the effects. Mitigation strategies may include utilizing multi-path routing protocols or using protocols that identify and eliminate malicious nodes from route caches or future route discovery.

Multi-path routing algorithms that produce \( k \) node-disjoint routes provide communication in the presence of up to \( k-1 \) Byzantine adversaries [32,33]. Once multiple paths are established, communication can be distributed via load balancing, duplicate packets can be sent over multiple routes for redundancy, or a single path can be used and the initiator can change to an alternate backup path only when malicious activity is detected on the primary path. Approaches using multi-path principles include the Secure Message Transmission (SMT) [34], Secure Multipath Routing (SecMR) [33], and the AdaptivePath [32] protocols.

Another mitigation strategy for attacks against the data forwarding phase is to monitor links for malicious activity and eliminate or avoid nodes exhibiting malicious behavior. Methods utilizing this approach include the watchdog-pathrater [35], On-Demand Secure Byzantine Routing (ODSBR) [22], CORE [36], and CONFIDANT [37,38] protocols. Additional concepts proposed to protect against misbehaving nodes include random two-hop acknowledgments [39], iterative and unambiguous probing mechanisms [40], and signed tokens [41].
2.3 Chapter Summary

In this chapter we discussed necessary background over ad hoc routing fundamentals and security issues faced by MANET routing protocols. Section 2.1 covered ad hoc routing operation, introduced a single-phase and two-phased routing classification, and reviewed foundational MANET routing protocols. Section 2.2 discussed the security issues related to the MANET routing process. We introduced a security definition based on if a routing protocol meets its intended goals. We also discussed proposed security solutions to protect both the route discovery and data forwarding phases.
CHAPTER 3
CURRENT EVALUATION TECHNIQUES

This chapter surveys current evaluation techniques being used to analyze security properties in MANET routing protocols. This chapter is based on topics and ideas generated in our previous works [7,42].

To fully understand MANET protocol security properties, routing protocols must be comprehensively evaluated to ensure that accuracy and reliability are maintained in malicious environments. Unfortunately, no comprehensive evaluation process is available to analyze MANET routing protocol security. Most MANET routing protocol literature includes haphazard security analysis, if any.

Table 1 lists some security evaluation approaches that have been used to analyze MANET routing protocol security. We categorize the evaluation approaches into two classes: non-exhaustive and exhaustive. Non-exhaustive approaches do not follow a comprehensive process, resulting in unrepeatable or inconsistent outcomes. Exhaustive techniques follow a structured, comprehensive approach resulting in higher confidence. However, exhaustive approaches are difficult to put into practice.

Table 1. Security Evaluation Approaches

<table>
<thead>
<tr>
<th>Analysis Approach</th>
<th>Approach Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unknown Attacks</td>
</tr>
<tr>
<td>Non-exhaustive</td>
<td>Visual Inspection</td>
</tr>
<tr>
<td></td>
<td>Network Simulation</td>
</tr>
<tr>
<td>Exhaustive</td>
<td>Analytical Proof</td>
</tr>
<tr>
<td></td>
<td>Simulatability Models</td>
</tr>
<tr>
<td></td>
<td>Formal Methods</td>
</tr>
</tbody>
</table>

It is crucial to distinguish each security evaluation approach to determine the type of information each provides and to determine where a technique should be used. Table 1’s first characteristic, unknown attacks, determines if an approach can discover unknown attacks or if it is limited to reporting known attacks. For instance, an attack must be known a priori in order to replicate in a network simulator. Conversely, human intuition provided by the visual inspection technique can uncover previously undiscovered attacks. The second characteristic, property guarantees, examines if an analysis approach can provide property guarantees. That is, given a security property, can the analysis approach guarantee that a property holds or not? Exhaustive evaluation techniques either prove a property exists or evaluate all possibility outcomes for the
security property in the given scenario. The final characteristic, *unconditional security*, is not attainable. Although certain approaches can guarantee security properties within scenario boundaries, we cannot analyze if a protocol is secure against any yet undiscovered attack or in unrestricted threat environments.

In the remainder of this chapter, we present our foundational work addressing how the analysis approaches listed in Table 1 are currently being used to evaluate security in MANET routing protocols.

### 3.1 Non-Exhaustive Evaluation Approaches

Non-exhaustive techniques cannot comprehensively evaluate a given protocol scenario nor can they guarantee that a security property holds in all situations. Nevertheless, visual inspection and network simulation can provide valuable information.

#### 3.1.1 Visual Inspection

MANET security analysis is commonly presented as a broad discussion utilizing a manual visual inspection process. Visual inspection is the oldest analysis technique, as it is in essence a variation of the *structured walkthrough* approach used in software engineering practices [42]. Visual inspection draws from human analysis and intuition, frequently used by a single analyst or used in interactive discussions within a small research group. Human intuition cannot provide exhaustive analysis over all possible attacker scenarios. In spite of its non-comprehensive process, visual inspection is the most commonly used approach to evaluate security properties in MANET routing protocols due to its simplicity. Visual inspection allows one to readily identify glaring security flaws, but lacks in the ability to exhaustively ensure attacks do not exist.

Figure 5 portrays how visual inspection progresses towards an iterative cycle of publishing a *secure* protocol, followed by publications attacking the protocol, which drives protocol adaptations and back again to attack analysis. This cycle solicits peer feedback and may find overlooked vulnerabilities, as research thrives though peer interaction. However, visual inspection cannot guarantee secure solutions, since many protocols claiming to be secure result in insecure systems.

We illustrate the visual inspection process according to the following development cycle: $DSR \rightarrow SRP \rightarrow Ariadne$. Looking back to Figure 3 on page 7, DSR trusts each node to correctly append its own identification into the path during route discovery. The DSR development process [14] focused on route discovery and performance, assuming all nodes followed the protocol rules. In a malicious environment, an adversary can simply relay a message without adding itself to the path or can inject false routing information to corrupt the path. These security failures render DSR susceptible to malicious activity and question the legitimacy of the returned routes.
SRP [23] extends DSR, attempting to provide end-to-end security via an existing security association between the source and destination. The destination uses the security association to compute a keyed message authentication code (MAC) over the received path sequence. The MAC is transmitted in the route reply, allowing the source to confirm the discovered route has been certified by the destination. Papadimitratos and Haas use visual inspection to discuss possible attacks and conclude that SRP is secure if adversaries do not collude or work together. The authors also use the formal method BAN logic [43] to claim SRP is always secure against single, non-colluding attackers.

Marshall et al. [44] show the SRP BAN analysis is false, illustrating an attack in which a malicious node $M$ simply does not add itself to the accumulated route path during route discovery and subsequently relays the signed route reply$^5$. The source accepts the validated corrupt path without the required node $M$, since the destination signed the corrupt path it believed was valid. They additionally call attention to the fact that BAN analysis should not have been used in this case, as the BAN authors themselves [43] admit BAN logic was developed to analyze beliefs between trusted nodes and was not intended to analyze security in malicious environments. An additional attack in which a single node actively injects false information into the SRP path is presented in [21]. The authors expose the attack using visual inspection after simulatability models revealed SRP is not provably secure. Simulatability models are further discussed below in Section 3.2.2.

Extending SRP's end-to-end security attempt, Ariadne [24] attempts to provide hop-to-hop security by requiring authentication at each intermediate node. During route discovery, each intermediate node digitally signs and appends a per-hop one-way hash value to the route request. Once the destination receives the $rreq$, it uses its security association to ensure the request is from the claimed source. Once validated, the target digitally signs the embedded route and returns the signed route to the initiator in the route reply. The original Ariadne security analysis relies on visual inspection in various attacker scenarios. Buttyán, Vajda, and Ács [21,45]

$^5$ Commonly called an invisible node attack.
discover multiple attacks against Ariadne. These attacks were investigated using visual inspection after their simulatability models indicated Ariadne is not provably secure.

Tying the discussion back to Table 1, visual inspection can successfully discover unknown attacks against published protocols. However, visual inspection does not offer a complete solution, since it cannot guarantee that a given attack does not exist. In addition to SRP and Ariadne, ad hoc routing protocol developments relying on visual inspection for security analysis include SEAD [26], SAODV [27], ARAN [28], SPAAR [29], and SecMR [33].

3.1.2 Network Simulation

Network simulation packages, such as OPNET, Network Simulator (NS-2), and Global Mobile Information Systems Simulation Library (GloMoSim), are integral to the MANET development community. Careful and detailed simulation has the potential to accurately approximate network performance [46]. However, Table 1 indicates network simulation cannot identify unknown attacks nor can it be used to exhaustively guarantee security properties. Attacks realized in network simulators must be known \textit{a priori} in order to code precise attacker actions into the simulation package. Furthermore, network simulation packages are primarily used to estimate network performance, based on statistical data analysis over independent runs. It is difficult to use network simulation to determine if a security property holds in all situations, since malicious activity is not necessarily statistical in nature. Attacks provide no clear statistical distribution. Are attacks random or do they occur at a uniform rate? Attack likelihood does not indicate if a protocol is secure against the attack.

Network simulation is generally useful in evaluating MANET security enhancements to the route discovery phase. For instance, the SEAD [26] development used simulation to estimate the increased overhead as compared to DSDV. Simulation was not used to evaluate SEAD’s security enhancements against malicious intent. Network simulation performance estimates evaluating MANET routing protocols are typically isolated to proposed enhancements intended to secure the data forwarding phase. Simulating data forwarding security enhancements reflects the resulting performance effects, such as changes in overhead and deliverability statistics, but cannot capture security properties.

We use the ODSBR protocol [22,47] development to demonstrate simulation's current use to evaluate security properties intended for the data forwarding phase in MANET routing protocols. ODSBR attempts to eliminate Byzantine attacks against the data forwarding phase by initiating a probing process once packet deliverability falls below a specified threshold. Awerbuch et al. use simulation to study performance differences between ODSBR and AODV under a range of Byzantine attacks, while varying the number of attackers. Their simulation data indicates that ODSBR is less susceptible to packet loss than AODV under these attacks, since the ODSBR probing routine identifies and eliminates malicious nodes once they stop following the protocol.
rules. Their simulation also suggests that malicious nodes located near the network center increase the attack success rate.

Based solely on the simulation results, the authors claim ODSBR is more secure than AODV. We agree their reported simulation data indicates ODSBR offers increased packet deliverability over AODV when analyzed against their attack scenarios. However, the authors make an extremely strong statement. We cannot infer from the their simulation data how the protocol reacts in different scenarios. For instance, the discussed ODSBR simulation assumes once a node stops forwarding packets it continues operating maliciously during the probing phase. What if the attacker cooperates in the probing phase and is not identified as malicious, will ODSBR still outperform AODV? The authors’ simulation data does not address this scenario. If we intuitively examine the scenario with the visual inspection process, the added overhead from ODSBR’s probing phase to unsuccessfullly attempt to discover the now properly operating malicious node results in ODSBR having a decreased packet deliverability than AODV.

Data-forwarding security-enhanced protocols that rely on simulation to study performance effects under various attacker strengths and configurations include SMT [34], watchdog-pathrater [35], CONFIDANT [37], random two-hop acknowledgments [39], and the iterative and unambiguous probing mechanisms used by Kargl et al. [40] to detect malicious nodes. Similar to the above ODSBR example, all these simulations estimate the performance effects of the security enhancements when measured against their un-enhanced counterparts, measuring either deliverability or the failure detection rate. The simulation data contained in these comparison studies imply that, for the given scenarios, the security enhancements are more resilient to malicious nodes that decide to drop packets during the data forwarding phase without directly studying how these mechanisms themselves can be subverted.

Although network simulation can identify protocol overhead costs and provide valuable insight into how a protocol may mitigate certain attacks, it tells us very little about any protocol’s overall security properties. Simulation cannot supply an exhaustive approach to claim protocol security, as simulation is being used to estimate the resulting performance effects and not directly being used to evaluate attack success against the proposed security mechanisms. Additionally, simulation-based performance studies can produce misleading results due to different simulation packages [48], using different physical layer modeling of wireless radio signals [49], incomplete documentation, and improper or missing statistical analysis procedures [50].

As we saw in the example attack on ODSBR’s probing process, simulation needs to be complemented with another evaluation technique, such as visual inspection, to consider attack avenues directly against the proposed security enhancement.
3.2 Exhaustive Evaluation Approaches

As indicated in Table 1, the exhaustive nature inherent in analytical proofs, simulatability models, and formal methods can guarantee that stated security properties can be met for a given scenario. If a security property is not proven, then an attack may exist, allowing exhaustive approaches the ability to discover unknown attacks. In formal methods approaches employing model checking, proof failures produce a protocol counterexample, directly illustrating the respective attack. In the remainder of this section we present how exhaustive evaluation approaches are currently being used to analyze MANET security properties.

3.2.1 Analytical Proof

Mathematical proof systems apply properties, theorems, and lemmas to prove or disprove specified security goals. Analytical proofs also provide the ability to validate the formal methods techniques.

As an example, Burmester, Le, and Yasinsac [11] introduce the cell-grid approach, enabling mathematical properties to prove efficiency and security properties for various gossip protocols. Gossip protocols can be used as a single-phased routing approach, with the goal to deliver maximum node coverage without suffering from the overhead resulting from message flooding. Single-phased routing protocols must ensure that message deliverability is guaranteed, as long as at least one non-malicious route exits to a given node. The authors analyze six gossip based protocols using a cell-grid overlay, as shown in Figure 6 (adapted from [11]).

![Figure 6. Cell Grid Overlay](image)

Using cell analysis to determine node retransmission reduces flooding’s node-to-node retransmissions to cell-to-cell retransmissions. Thus, only one node within a cell must retransmit a given message. The cell size is calculated such that the furthest distance between any two points in neighboring cells is at most one hop, as indicated by the dotted line. The authors use the cell-grid overlay to analytically prove the deliverability coverage probability for the six gossip protocols, to include two protocols which they prove 100% coverage in environments that include malicious threats. For instance, the authors introduce a dynamic cell-based gossip protocol in
which each node (x) retransmits its first reception of message \( m \) with the probability \( p_x = \frac{5\pi}{k} \frac{k}{n_x} \), where \( 5\pi \) is based on the optimal cell area\(^6\), \( n_x \) is node x’s 1-hop neighborhood density, and \( k \) is a tunable propagation parameter adjusted to achieve the desired retransmission probability vs. the desired coverage probability.

The example above illustrates that the cell-grid approach exhaustively guarantees via proof the single-phased routing protocol deliverability requirement. However, it is difficult to capture two-phased route discovery interactions using manual proofs. Proof complexity increases as the system size enlarges and may not easily take into account all message interactions.

In addition to lacking automation, mathematical proof systems are exceedingly reliant on system assumptions and the approach can differ greatly, based on the researcher’s individual mathematical capabilities or preferences. Additionally, analytical proofs cannot be easily formalized into a general process. We cannot simply follow a set of steps for all situations. Each proof must be customized to the individual scenario, such as the previous cell-grid example. Tailoring proofs to an individual problem can differ significantly, based largely on the analyst’s individual preferences. The greater dependence on human intellect and manual interaction increases the possibility of unintentional error injection, which can result in multiple or ambiguous results.

3.2.2 Simulatability Models

Simulatability models have been recently proposed to analyze route security in MANETs. The simulatability approach parallels a widespread technique traditionally used to develop cryptographic proof systems for authentication protocol evaluation [51,52,53]. A protocol is considered secure using simulatability models if a real protocol can be abstracted into, or simulated by, an ideal protocol model such that an adversary has no greater capability against the real protocol than it does against the abstracted ideal protocol. The ideal-world abstraction uses a theoretical oracle, or trusted third party, that has complete system knowledge. The oracle knowledge renders the adversary powerless against the ideal protocol. For example, a MANET node with oracle knowledge over the complete network topology does not accept corrupted routes that are not topology consistent. Unfortunately, real-world MANET nodes do not have oracle knowledge and cannot maintain a complete trusted network connectivity view.

To the best of our knowledge, Buttyán, Vajda, and Ács offer the only research using formal simulatability models to evaluate provable routing security in on-demand source routing protocols, providing simulatability security analysis over DSR, SRP, and Ariadne [21,45]. In [54], they extend their model to analyze on-demand distance vector protocols, to include analysis over AODV, SAODV, and ARAN. The authors model the real and ideal protocol abstractions according

\(^6\) Each cell is approximately \( \frac{1}{5\pi} \) of the entire hop circle based on the ideal edge length.
to Figure 7 (adapted from [45]), using Turing machines to theoretically simulate a protocol at an instance in time, with no mobility. The only distinction between the ideal and real world model is that in the ideal case, the shaded area is controlled by machine T. Machine T acts as an oracle to validate routes against the complete ad hoc network connectivity defined by the undirected graph $G = (V, E)$, where $V$ represents the set of nodes and $E$ is the set of network links between nodes. Non-malicious nodes are modeled by machines $M_1 – M_n$, adversarial nodes are modeled by machines $A_1 – A_m$. Machine C models the wireless communication process by delivering each node’s output to the respective neighbors that are identified by the network connections stored in graph $G$.

![Figure 7. Simulatability Modeling](image)

Machine H is responsible for initiating new route requests and receiving all possible routes. The routes are represented by $\text{Out}_{\text{real}}$ or $\text{Out}_{\text{ideal}}$. For the ideal protocol abstraction, the oracle checks to ensure the route is consistent with the complete network knowledge of $G$, dropping corrupt routes from $\text{Out}_{\text{ideal}}$. A protocol is considered secure in their model if all returned routes are valid possibilities according to the connectivity graph $G$. Consequently, a protocol is considered provably secure if $\text{Out}_{\text{real}} = \text{Out}_{\text{ideal}}$.

In [21,45], Buttyán, Vajda, and Ács exercise their simulatability model to discover that SRP and Ariadne are vulnerable to attacks against the route discovery phase. They also use simulatability models to claim that their proposed endairA protocol is provably secure. The authors extend their work in [54] to evaluate on-demand distance vector protocols, providing an attack on SAODV and using their extension to claim that ARAN is provably secure. Provable security claims over the route discovery process does not guarantee that a Byzantine adversary may not be included in a valid path, since a malicious insider may follow the protocol rules during route discovery and subsequently choose to drop packets during the data forwarding phase.

While the simulatability model used in [21,45,54] can provide provability claims, the model cannot guarantee security in all cases, since the model is limited by the environmental and attacker assumptions. This issue highlights the distinction between provable and unconditional security.
security. While a provably secure model holds for its given assumptions, attacks may still exist against the protocol in unrestricted threat environments. While all modeling techniques require assumptions to bind the problem to its context, we cannot presume that a malicious attacker follows these restrictions.

Following the unconditional security concept, we give, in [55], attacks against endairA and ARAN through the intuitive power inherent in the visual inspection technique. The proposed simulatability model was originally intended to avoid overlooking subtle flaws commonly missed by non-exhaustive analysis techniques. Buttyán, Vajda, and Ács claim that if the ideal protocol model is shown to never encounter and eliminate a corrupt route, then Out_{real} = Out_{ideal} is always true and the protocol is provably secure. They claim endairA and ARAN are provably secure based on their static analysis that no attacker against the ideal cases can ever force the oracle to drop routes inconsistent with the network topology.

Conversely, we show how the authors’ dismissal of the Sybil\footnote{A Sybil attack allows an attacker to claim multiple identities.} [56] attack leads non-malicious nodes into believing an incorrect network view. Buttyán, Vajda, and Ács argue the Sybil attack cannot occur, since nodes authenticate their neighbors during neighborhood discovery. However, their model allows for colluding adversarial nodes. Colluding attackers can share digital signature keys, allowing an attacker to masquerade as multiple nodes, even during neighborhood discovery. The attacks we present in [55] leverage this improper assumption using the network connectivity shown in Figure 8, where shaded nodes signify corrupt nodes, solid lines signify actual connectivity reflected in network graph $G$, and dashed lines represent spoofed links. In the real protocol model, the routes $S$-$Y$-$D$ and $S$-$Z$-$D$ exist, while the ideal model shows the actual network connectivity does not contain these paths. The disagreement over these two paths between the ideal and real models result in Out_{real} $\neq$ Out_{ideal}, thus the protocols are not provably secure against colluding attackers in an unconditional environment.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{fig8.pdf}
\caption{Real vs. Ideal Network Connectivity}
\end{figure}

While we have illustrated in the above example the misconception of provable security in MANET routing protocols, we point out an additional limitation to the simulatability model.
Exercising the simulatability model may lead to the misconception that simulatability models provide an automated analysis capability. The Turing machine simulation provides a theoretical basis for Buttyán et. al's manual analysis. In all cases where the authors' model indicated a protocol is not secure, visual inspection was used to discover the actual attack sequence.

3.2.3 Formal Methods

Formal methods are another exhaustive technique which has yet to see extensive use to analyze security properties in MANET routing protocols. Formal methods grew out of the software engineering field as a tool to increase integrity and reliability in safety-critical systems, such as flight-control systems or nuclear-plant operations [57,58]. Although formal methods cannot provide unconditional system correctness guarantees in unrestricted environments, their use is beneficial since they can significantly increase system confidence.

In formal methods approaches, a system and its desired properties are formally specified in strict mathematical representation. Once the system is formally defined, model checking or theorem proving techniques can be used to evaluate the system against the desired properties. Theorem proving results in formal proofs using the semantics and axioms, or rules, specified by the given formal method. However, similar to analytical proofs, theorem proving may become complex for large systems.

The approach taken by model checkers is to construct a finite system model and exhaustively search all achievable states to determine if the evaluated property, such as an insecure state, occurs [59]. Regrettably, model checking may not be able to analyze large or complex system representations, since they suffer from state-space explosion. There has been significant research focuses into state-pruning techniques, symbolic representation, partial order reduction, and state compression to combat the state-space explosion problem [60,61,62,63].

Tying the discussion to security analysis, formal methods have shown success in evaluating authentication protocols, such as the Needham-Schroeder Public Key (NSPK) protocol and its variants [64,65]. It is vital that we differentiate between authentication protocols and MANET secure routing protocols. The primary goal for an authentication protocol is to securely share information, such as a session key, between two nodes. Security analysis for authentication protocols evaluates if it is possible for an adversary to obtain access to the protected key, regardless of intermediate nodes within the communication path [66]. Conversely, security evaluations for MANET secure routing protocols must consider actions taken by intermediate nodes. We must consider whether the intermediate nodes can impact the secure routing protocol's intended goal. More specifically, we must consider route accuracy to secure the route discovery phase and protocol reliability to secure the data forwarding phase.

Authentication protocols are commonly referred to as security or cryptographic protocols.
As the primary thesis objectives focus on procedures to evaluate security properties in MANET routing protocols, Table 2 lists formal methods which have been used to analyze MANET secure routing properties.

Table 2. Formal Methods used in MANET Security Analysis

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadcast Calculus [67,68]</td>
<td>Uses process algebra to model channel interactions</td>
</tr>
<tr>
<td>Cryptographic Protocol Analysis Language Evaluation System (CPAL-ES) [69]</td>
<td>Uses weakest precondition logic to generate required verification conditions</td>
</tr>
<tr>
<td>Petri Nets [70,71]</td>
<td>Graphically models distributed information flows</td>
</tr>
<tr>
<td>Strand Spaces [72]</td>
<td>Graphical representation of causal interaction</td>
</tr>
</tbody>
</table>

**Broadcast calculus.** Figure 9 illustrates the relationship between various calculus, or process algebra, formal modeling languages that provide the ability to evaluate MANET route security analysis with broadcast calculus. Pi calculus [73] models channelized point-to-point protocol interactions using the Calculus of Communicating Systems (CCS) process algebra [74]. Spi calculus [75] extends pi calculus by integrating cryptographic primitives to support authentication protocol modeling and analysis.

Following the process provided by the spi extension to pi calculus, Nanz and Hankin [67,68] extend broadcast calculus to analyze MANET route security. Their core broadcast calculus language is the Calculus of Broadcast Systems (CBS) [76]. The CCS based pi and spi calculi model communication via point-to-point channels. CBS based calculi provides the ability to model broadcast communication. The capability to model broadcast communication enables CBS to model local area network (LAN) segments, in which all nodes receive the communicated message at the same time. Nanz and Hankin extend CBS into CBS#, incorporating cryptographic primitives, similar to spi calculus, and implement local broadcast communication. Local broadcast supports wireless communication modeling, since local broadcast messages are restricted to neighboring nodes within a node’s transmission footprint.

![Figure 9. Calculus Modeling Relationships](image-url)
Nanz and Hankin define secure route discovery as *topology consistency*. A MANET routing protocol maintains topology consistency as long as each node’s routing tables do not contradict the actual network topology. The authors illustrate their technique by demonstrating a malicious insider can generate false routes by injecting topology inconsistencies into SAODV for the network topology shown in Figure 10.

![Figure 10. Topology Inconsistency Attack](image)

Once SAODV is specified within the CBS# process algebra, Nanz and Hankin employ control flow analysis to approximate possible protocol outcomes. Control flow analysis techniques [77] predict possible value sets for process outcomes. Table 3 provides the control flow analysis results on the CBS# defined SAODV, indicting an SAODV attacker can inject topology consistency failures. The failures contradict the true network topology from Figure 10 due to the malicious node $n_a$’s ability to spoof its sending address as either $n_1$ or $n_2$.

<table>
<thead>
<tr>
<th>Node $n_1$:</th>
<th>Destination</th>
<th>Next-hop</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n_2$</td>
<td>$n_1$</td>
</tr>
<tr>
<td></td>
<td>$n_2$</td>
<td>$n_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node $n_2$:</th>
<th>Destination</th>
<th>Next-hop</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n_1$</td>
<td>$n_2$</td>
</tr>
<tr>
<td></td>
<td>$n_1$</td>
<td>$n_1$</td>
</tr>
</tbody>
</table>

While the analysis performed by Nanz and Hankin is based on the static control flow analysis procedures, they provide control flow automation via constraint solvers. Constraint solvers search for the resolution of a system’s desired information or stated goals [68]. The authors results are restricted to the SAODV and ARAN ad hoc distance vector protocols and they analyze limited network topologies.

**Petri nets.** Petri nets [70,71] were introduced by C.A. Petri in 1962 as a graphical representation to model information flow in concurrent or asynchronous systems. Petri nets model the information flow between places, or states, using transitions which are governed by *firing* rules. For example, the Petri net in Figure 11 consists of three places and one transition. The black dots in Figure 11a indicate that places $P_1$ and $P_2$ have tokens, enabling the transition $T_1$ to fire. After the transition is fired, the tokens in $P_1$ and $P_2$ are consumed and a token is produced in $P_3$, as shown in Figure 11b. Coloured Petri nets (CPNs) [78] extend traditional Petri nets by assigning value, or labels, to the initial binary token model.
Petri net models can analyze reachability, providing the ability to prove correctness properties. Within the cryptographic proof community, Petri nets and CPNs have been used to evaluate authentication protocol security in [79,80,81,82]. In the MANET routing community, CPNs have been used to show protocol correctness (in [83]) for AODV and to study position based protocol adaptations (in [84]) to the On-Demand Multicast Routing Protocol (ODMRP) [85].

Similar to network simulation, the Petri net focus for evaluating MANET security properties lies within the data forwarding phase. However, unlike the simulation result expressed as statistical performance, the Petri net evaluation in [86] produces proofs over the data forwarding mechanism. Djenouri and Badache use Petri nets to evaluate security in a secure data forwarding protocol. The authors model their two-hop acknowledgement scheme to detect selfish nodes that refuse to forward packets during the data forwarding phase. Their Petri net analysis shows their protocol successfully detects all maliciously dropped packets when the analysis environment is isolated from non-malicious failures.

**CPAL-ES.** The Cryptographic Protocol Analysis Language Evaluation System (CPAL-ES) was initially developed to formally analyze authentication protocols [69,87]. CPAL-ES is based on Dijkstra’s formal reasoning called weakest precondition [88]. Weakest precondition is an extension of original logic introduced by Hoare [89]. In its most basic form, weakest precondition reasoning is stated as:

\[
P \{ A \} G,
\]

where \( P \) is the precondition, \( A \) specifies the protocol actions, and \( G \) specifies the protocol goals or post conditions. Weakest precondition logic can also be characterized mathematically as \( y = f(x) \), where \( y \) represents the post conditions, \( f() \) represents the protocol actions, and \( x \) depicts the precondition.

Figure 12 illustrates the CPAL-ES analysis process. The first protocol analysis step is to specify the protocol actions and security goals indicated by the shaded area. Protocol actions are represented in CPAL with operators such as send, receive, encrypt, and sign. The CPAL assume operator specifies the environmental assumptions. The CPAL assert operator specifies the protocol’s security goals. Once the protocol actions, goals, and assumptions are specified in CPAL, CPAL-ES generates the weakest precondition, or verification condition, required to make
the assertion true. The final analysis step is to prove the weakest precondition is true or show that it is not true. CPAL-ES has successfully analyzed numerous authentication protocols [69,87,90].

![Figure 12. CPAL-ES Protocol Evaluation](image)

In [91], Marshall uses CPAL-ES to evaluate SRP route security to ensure routes returned to the initiator exist in the network topology. His work is a first step towards automating analysis attempts to identify MANET route discovery security failures in on-demand source routing protocols. Instead of showing that two nodes exclusively share a secret, as in previous CPAL-ES authentication protocol evaluations, SRP is analyzed with CPAL-ES to determine if a malicious node can force the source into accepting a non-existent route to its intended destination. Marshall specifies the configuration shown in Figure 13 into CPAL. The attacker node I does not append itself to the route path as the SRP protocol expects. CPAL-ES analysis demonstrates the protocol accepts the route $S$-$A$-$B$-$T$ as valid, even though it is visually evident the route between $S$ and $T$ requires node $I$ in this configuration. His analysis follows proof by contradiction, in that the CPAL-ES analysis recognizes that SRP accepts a route that does not exist in the corresponding network topology. The CPAL analysis demonstrates that SRP is not secure against non-colluding attackers as initially claimed in [23]. Marshall's work shows how a formal verification system can be used to evaluate MANET routing protocol security properties. However, Marshall's work does not provide full automation. His SRP analysis is limited to the isolated network topology in Figure 13 and the protocol messages are hard coded to match the given configuration.

![Figure 13. SRP CPAL Attack Scenario](image)

**Strand Spaces/Athena.** The Strand Space Model (SSM) [72], was developed to analyze authentication protocol security. A *strand space*, portrays causal interaction for event sequences as a graphical representation. As an example, Figure 14 illustrates a strand space containing the following components:
- **Strand**: \( \downarrow \) represents the event sequence for a single node.
- **Bundle**: A strand space portion containing at least one complete protocol instance.
- **Message**: \( \rightarrow \) represents information sent between node strands.

The strand bundle in Figure 14a represents the original NSPK protocol [64]. The protocol’s goal is to exclusively establish a session key between nodes A and B using the random nonces \( N_a \) and \( N_b \). Each message is secured with the intended recipient’s public key. The strand space bundle in Figure 14b duplicates Lowe’s [65] original NSPK attack. The attack occurs when node A attempts to establish a session key with a malicious insider node I. Node I uses the nonce \( N_a \) obtained from node A’s key establishment request and initiates a separate session to node B, while claiming to be node A. Node B believes it is communicating with node A and returns its nonce \( N_b \) encrypted with A’s public key. Node I cannot directly read the encrypted message, however, node I uses A as an oracle to decrypt \( N_b \) and completes the session with node B. The attack results in node B believing it has an authenticated session with node A, and B may unknowingly transmit secure information to node I, which I was not authorized to receive. When using the SSM to show a protocol is secure, manual proofs are constructed to show the causal relationships do not contain flaws.

Song, Berezin, and Perrig [92] developed the tool Athena to provide automation to the SSM analysis process. Athena analysis starts by specifying the security goal as the initial state and builds a proof tree according to the SSM inference rules. Athena eliminates state-space explosion, since it uses symbolic representation and free variables that do not require explicit processes for each node, and utilizes a backward search to reduce the possible search space. While Athena is not affected by state-space explosion, it is not guaranteed to terminate. If the Athena evaluation terminates properly, it results in a proof for the given security property or provides a counterexample showing the attack if the security property does not hold.

Following the SSM and Athena model, Yang and Baras [93] propose extensions to model security properties in ad hoc routing protocols. Since the original SSM models authentication
protocols using a fixed event message sequence, the extensions include branch support to model ad hoc routing decisions. The authors use their model to show SAODV cannot protect against an authenticated malicious insider’s ability to corrupt the routing path. Yang and Baras define security in terms of \textit{liveness} and \textit{safety}. Liveness is a protocol correctness property, stating that the protocol discovers routes if they exist. Safety implies that discovered routes are stable, indicating the routes have not been corrupted.

Since the Athena source code is not publicly available, Yang and Baras use their SSM extensions and generate customized code to implement Athena’s published search procedures. The automated search procedure produces a safety failure against SAODV, indicating a successful routing attack. The attack, using the network topology shown in Figure 15, tricks the source $S$ into believing that the path $S-I-D$ exists through the intruder $I$ to the destination $D$. To accomplish the attack, the malicious insider $I$ computes and sends the identical response as the valid node $B$. The attack is feasible since SAODV uses next-hop tables, relying on each node to calculate a hash value based on the current hop count. Consequently, nodes at the same hop distance can compute the same hash value. While the attack seems obvious in a visual inspection attempt, the authors’ research goal is to demonstrate automating such logic.

![Figure 15. SAODV Attack Analysis](image)

3.3 Chapter Summary

In this chapter we surveyed the existing approaches used to analyze security properties in MANET routing protocols. Section 3.1 focused on non-exhaustive evaluation approaches. Visual inspection has the power to leverage human intuition, resulting in many discovered attacks. Unfortunately, visual inspection cannot exhaustively determine if an attack does not exist.

Network simulation primarily concerns itself with statistical performance analysis. While network simulation lends well to estimate operational characteristics, it does not provide the ability to exhaustively determine if an attack exists. Network simulation’s role in MANET security analysis compares deliverability and overhead statistics for proposed security mechanisms as opposed to directly studying the proposed security mechanism’s vulnerabilities.

In Section 3.2 we focused on the current exhaustive MANET security evaluation approaches. We discussed how analytical proofs can guarantee message deliverability for isolated instances, but the process must be tailored to individual problems and it is difficult to analyze large systems.
with complex message interactions. The analytical approach’s reliance on human interaction may inject errors into the analysis and does not provide for an automated analysis capability.

The simulatability model attempted to claim provable security. However, we saw that the imposed limitations resulted in provably secure protocols that were susceptible to malicious activity. The simulatability model relies on executing a theoretical Turing machine and does not provide automated security analysis. Additionally, the simulatability model provides proofs that indicate an attack is possible. Unfortunately, simulatability models do not automatically identify the attack. Once the model indicates the protocol is not secure, the attack must be discovered via other means, such as visual inspection.

The formal methods approach provides the ability to formally specify MANET protocols and develop formal proofs or provide automated analysis through model checking. Formal methods adaptations to evaluate MANET routing protocol security properties is in its initial stages and has not seen widespread use. Current research demonstrates the feasibility to automate such modeling, but is currently restricted to isolated network topologies and instances. The most evolved work is presented by Nanz and Hankin [68]. However, their work is isolated to on-demand distance vector protocols and evaluates limited network topologies.
CHAPTER 4

BOUNDING THE EVALUATION

In this chapter we set the evaluation boundaries for the context of this thesis. The boundaries are not intended to limit the development and analysis capabilities, but are required to scope the research efforts. Section 4.1 formulates a comprehensive security evaluation framework. Section 4.2 sets the research focus onto evaluating the route discovery process and the protocol class we investigate. In Section 4.3, we isolate malicious failures, allowing us to attribute routing failures to malicious activity and to provide a reduced state representation required for exhaustive model checking analysis.

4.1 Towards a Comprehensive Security Evaluation Framework

MANET security evaluations within current literature primarily focus on a single evaluation approach and limit their analysis through restricted operational assumptions. While the attack possibilities discovered by the simulatability model [21,45,54] were followed by visual inspection to find scenarios where attacks were possible, visual inspection itself was not used as an independent analysis technique. We contend that the analysis techniques currently being used to evaluate security in MANET routing protocols have different, yet often complimentary properties.

The visual inspection process benefits from unrestricted human intuition, offering a useful approach to discover protocol attacks. However, the visual inspection iterative cycle illustrated in Figure 5 on page 14 cannot prove security properties exist for all cases, as new attacks are routinely discovered.

Network simulation packages are primarily used to project average case performance, based on statistical analysis over independent runs. While it is critical we understand the performance characteristics resulting from implemented security mechanisms, network simulation cannot identify undiscovered attacks nor can it be used to guarantee any posed security property. Attacks utilized in any network simulation must be known \textit{a priori} in order to code the attacker actions into the simulation package. Network simulation cannot be used to answer binary type questions since an attack possibility is not statistical in nature. Binary type questions result in a simple yes or no answer. The answer must be exhaustive for the given scenario. For instance, does the given security property hold, yes or no? Statistical network simulations that indicate an attack is not likely or do not identify an attack do not ensure a protocol is secure against the attack. Furthermore, the resulting performance effects may also be misleading due to variations between simulation packages and improper statistical analysis procedures [46].
Analytical proofs, simulatability models, and formal methods can determine if stated security properties can be met or not for a given scenario, determined by the attacker strength and network topology. In the case a security property is shown to fail, one can infer an attack exists, enabling the given analysis approach the ability to discover unknown attacks. In the formal methods approaches utilizing model checking, security failures produce a protocol counterexample illustrating the respective attack.

Based on these security analysis approach descriptions, we propose combing techniques to provide a comprehensive security analysis capability for MANET routing protocols. Since the route discovery and the data communication phases must be individually secured, as illustrated in Figure 4 on page 10, a comprehensive analysis framework must cover both phases. However, researchers commonly target only one individual phase. In Table 4 we propose a comprehensive security analysis framework to analyze security over a complete MANET routing solution. The framework can also be used to target an individual phase being specified by a given protocol development.

Table 4. Comprehensive Security Analysis Framework

<table>
<thead>
<tr>
<th>Targeted Phase</th>
<th>Security Analysis Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route Discovery</td>
<td>Visual Inspection</td>
</tr>
<tr>
<td></td>
<td>Formal Methods</td>
</tr>
<tr>
<td>Data Forwarding</td>
<td>Visual Inspection</td>
</tr>
<tr>
<td></td>
<td>Network Simulation</td>
</tr>
</tbody>
</table>

Evaluating data forwarding. We first look at the security analysis techniques for the data forwarding phase used once a route has been discovered. The goal in the data forwarding phase is to deliver data packets over an established route. Security mechanisms in this phase must first identify when packet delivery is not being maintained and mitigate the failure by choosing an alternative available path, notifying the route discovery phase to initiate a new route discovery process, or identifying malicious nodes to avoid in future paths. Packet deliverability thresholds attempt to differentiate between non-malicious and malicious failures. Since the attacks against this phase are not binary in nature, formal analysis techniques cannot be used to guarantee that a security property exists. Security mechanisms designed for the data communication phase are generally evaluated using network simulation and focus on performance criteria.

Network simulation, as it is currently being used, provides an avenue to study the operation and performance of the mechanisms being designed to secure the data forwarding phase. The question becomes, does the security enhancement provide a higher deliverability ratio than without any enhancement? However, the results are only applicable to the given attacker and network topology scenario simulated. According to the comprehensive security analysis framework in Table 4, we complement network simulation with the intuitive power inherent in the
visual inspection technique to find attacks that reduce the security enhancement's effectiveness against the data forwarding phase. For example, consider a protocol such as ODSBR [22], which attempts to eliminate malicious nodes by blacklisting them. What effects would a malicious node simply relaying data have within the network? What effect would a malicious insider node have that operates maliciously only against data forwarding packets and fully cooperates during probing activity attempting to identify the malicious nodes?

Evaluating route discovery. Analyzing security properties provided in the route discovery phase can benefit by combing visual inspection and formal methods, as indicated by the comprehensive security analysis framework in Table 4. The power inherent in the visual inspection approach provides an unrestricted look at route discovery. Many simple attacks, such as the invisible node attack [44,94] and the Sybil [56] attack, can be easily identified through visual inspection.

Complementing visual inspection with formal methods techniques allows us to search for attacks that actively corrupt the route discovery process. Since visual inspection cannot provide property guarantees, a formal approach must also be used. If a binary security property can be stated, a formal approach can evaluate if the property holds or not. For instance, if we evaluate route discovery over a static network for an instance in time, returned routes either exist or do not exist in the current network topology. However, analysis is currently limited by restricting attacker capabilities and by the specific network topology investigated.

We choose formal methods to complement the visual inspection technique to evaluate the route discovery process, since the analytical proof and simulatability exhaustive analysis techniques are not automated. Formal methods using model checking can provide automated analysis. However, model checking techniques must be adapted to evaluate MANET route discovery security properties before they can complete Table 4’s comprehensive security analysis framework. Chapter 6 provides research into automated model checking procedures to evaluate security in the MANET route discovery process.

4.2 Focusing on Route Discovery

As the primary research goal is to complete the proposed comprehensive security analysis framework by developing automated model checking techniques, we focus our research efforts on the MANET route discovery process. The goal in the route discovery phase is to determine a valid route from a source to a destination. Attacks against this process include altering a route so that it does not represent a valid path that provides confidence in the source, destination, and intermediate links.

While the data forwarding phase required analysis over a given time period to enable security mechanisms based on packet deliverability thresholds, attacks against the routes returned during the route discovery phase are binary for a specific instance in time. That is, for a
given static network topology, does the route discovery phase return a route that is consistent
within the representative connectivity graph? The connectivity graph contains the complete
network topology and is intended for modeling and analysis purposes only. The presence of a
connectivity graph for security analysis modeling does not imply that on-demand nodes within the
network have full network knowledge as required by full table-based protocols such as DSDV
[16].

We further concentrate our research focus to on-demand source routing protocols. This focus
allows us to readily determine route validity, since the selected path is explicitly included in the
route reply. Returned routes that are not possible in the network topology are considered invalid.
An attacker’s intent is to force the protocol to accept a route that does not exist and cannot
subsequently be used for data communication. Specifically, we model and evaluate the SRP,
Ariadne, and endairA protocols. These protocols are a natural progression to the initial insecure
DSR protocol. Selecting these protocols allows us to model a common baseline functionally
provided by DSR, extending the model to include the added security features provided the
proposed DSR extensions.

While the thesis research focuses on source initiated on-demand protocols, we can also apply
automated model checking to table-based on-demand protocols such as AODV. To check for
valid routes in table-based protocols, the model requires the ability to track and evaluate each
node’s local tables to validate each next-hop entry against the connectivity graph.

Nanz and Hankin [67,68] provide initial work to automate analysis in table-based protocol
evaluations. Their work provides automation to a few simple network topologies and does not
exhaustively analyze all network topologies for a given network size. Our work provides the first
automated exhaustive analysis approach to evaluate security properties in ad hoc on-demand
source routing protocols.

### 4.3 Isolating Malicious Failures

MANET routing protocols mitigate non-malicious failures via reliability mechanisms, such as
reinitiating route discovery or using multipath protocols. Non-malicious routing failures can occur
due to mobility or failed nodes, which affects a protocol’s message deliverability and overall
protocol performance. Malicious failures may also occur, which to the routing protocol cannot be
distinguished from non-malicious failures. In order to effectively model security properties in
MANET routing protocols, we isolate malicious failures during model development.

In addition to the inconclusive results encountered by including non-malicious failures,
modeling non-malicious failures may not be feasible in a finite space model checker. While model
checkers may encounter the same state-space limitations when isolating malicious failures,
modeling only the required failures of interest reduces the model complexity and increases the
chances that the model checker can evaluate the given security property.
As an example, consider model checking for a routing protocol that includes node mobility. Wibling et al. [95] evaluate routing protocol loop freedom and show that it is not feasible to model mobility in model checking paradigms due to rapid state-space explosion. In contrast, if it were feasible to model mobility, the cause of any routing inconsistencies are inconclusive. It is not possible to distinguish if routing failures in this modeling approach were due to malicious activity or due to link breakage resulting from node mobility.

Based on these observations, we choose to focus on modeling malicious failures against secure routing protocols. Eliminating non-malicious failures allows us to effectively isolate discovered route failures due to malicious activity. However, we must ensure that focusing solely on malicious faults does not alter a modeled protocol's security semantics. Removing non-malicious routing failures must not change or bias the protocol's security functionality or vulnerabilities.

To ensure eliminating non-malicious failures during model development does not alter the results, we adopt the precedent derived by the adversary advantage definitions presented in [96]. The adversary advantage is the attacker's success probability, where a protocol provides either $\Gamma$-availability or $\Gamma$-tolerance, if $\Gamma$ is the attacker node(s) within the network. A protocol maintains $\Gamma$-availability if the adversary advantage is negligible for all $\Gamma$-adversaries, which is considered perfect security. Perfect security is not attainable in MANETs, since attacks such as the invisible node attack [97] cannot be completely eliminated and undetected compromised insiders can drop packets at any time.

A routing protocol maintains $\Gamma$-tolerance if the adversary advantage - non-malicious protocol failure] is negligible. For instance, if a protocol works $X\%$ of the time without an adversary, does the presence of an adversary reduce $X\%$? Adopting the reasoning behind $\Gamma$-tolerance, we choose to model routing security features without including non-malicious failures. Therefore, failures discovered in our models are attributable only to malicious activity.

### 4.4 Chapter Summary

In this chapter we set the boundary conditions to define the thesis context. In Section 4.1 we introduced a comprehensive security evaluation framework, combining evaluation approaches to provide a complete security analysis capability. We saw that further research in formal methods techniques is required to complete the comprehensive framework. In Chapter 6 we develop an automated security evaluation approach following the model checking paradigm. Developing a formal methods approach completes the security evaluation framework and provides a comprehensive MANET security analysis capability.

Section 4.2 further refines the research to evaluate the route discovery process and focuses model development and analysis onto ad hoc on-demand source routing protocols. The protocols we chose for evaluation are based on the core DSR functionality and provide a natural security
progression, attempting to secure the explicit route embedded into source routing protocols. We saw that the route discovery process can be viewed as a binary analysis decision when we view the network as a snapshot, or instance, in time.

In Section 4.3 we isolate modeling malicious failures in order to reduce the associated state-space required during model checking development and to provide a static analysis picture. We saw that by removing non-malicious failures from the protocol models, route failures detected during analysis are solely attributed to malicious activity.

These boundaries guide the research for the adaptive threat modeling approach contained in Chapter 5 and the automated security analysis development provided in Chapter 6. These boundaries do not limit the research efforts, but allow us to provide a usable security evaluation process to identify protocol security flaws in ad hoc route discovery mechanisms.
CHAPTER 5

ADAPTIVE THREAT MODELING

Regardless of the selected security evaluation technique, analysis results are limited by the threat model used to analyze a given protocol. An inappropriate threat model is likely to produce incorrect results, possibly claiming protocol security when it indeed does not exist. In this chapter we develop an adaptive threat model, based on our previous work [98], to analyze attacks against the route discovery phase in MANET routing protocols. By using an adaptive threat model, we can identify the minimum attacker capabilities required to break MANET route discovery mechanisms.

Adaptive threat modeling is different than the traditional approach that limits, or bounds, the analysis results by placing restrictions on the attacker. In the traditional approach, authors are inclined to claim protocol security. However, the security results are only applicable if analyzed in relation to the author’s assumptions on the attacker capabilities. Author claimed secure routing protocols may not be secure when analyzed outside of the respective author’s assumptions. For instance, work in [3] and [4] lists many proposed ad hoc routing security solutions along with their operational requirements and drawbacks. Inconsistencies between anticipated threat models and attacker capabilities produce secure routing protocols that contain flaws and are susceptible to attacks. Additionally, secure routing protocols cannot be compared without common attacker assumptions or security definitions.

Adaptive threat models do not suffer from the artificial limitations resulting from bounded security evaluations based on restricted author assumptions. The adaptive threat model is intended to investigate MANET routing protocols for the existence of vulnerabilities, not the absence of such under an author’s declared restrictions. Since the attacker capabilities are not fixed, adaptive threat modeling allows for a common or baseline comparison between multiple protocols.

Developing an adaptive threat model is a foundational step required to support the exhaustive analysis procedures developed in Chapter 6. Threat models must be specified during model checking development. However current MANET threat models specify a limited attacker, not allowing an exhaustive evaluation over the analyzed protocol.

The remainder of this chapter is outlined as follows. In Section 5.1, we discuss attack sources and classical routing attacks. Section 5.2 includes the canonical approach to threat modeling and identifies the limitations faced by existing MANET threat models. Section 5.3 contains the adaptive threat model, illustrates the model over an example scenario, and classifies various current attacks in relation to the model.
5.1 Attacking Route Discovery

Following the evaluation focus specified in Chapter 4, we develop the adaptive threat model to focus on attacks against the route discovery phase. Route discovery attackers attempt to corrupt routes so they are inconsistent with the network topology.

5.1.1 Attack Sources

We consider two attack sources: outsider or insider.

- **Outsider** attackers do not have trusted keys. They typically rely on message relay, replay, or delay to influence routing protocols. Outsider attackers may also be able to inject false routing information into improperly designed routing protocols, although these outsiders cannot sign messages themselves.

- **Insider** threats occur when a trusted node, with appropriate keying material, is compromised. Malicious insiders may be able to inject corrupted, but legal messages into protocol instances.

Malicious insiders are much more difficult to defend against than malicious outsiders because they hold legitimate keys. Additionally, malicious insiders can generally act as malicious outsiders as well.

In addition to attempting to inject false routing information, malicious insiders can attack the data forwarding phase by dropping data communications on discovered routes in which they are included. Insider packet dropping cannot be eliminated, since the attacker may operate non-maliciously according to protocol rules until it decides to drop data packets. While the adaptive threat model and automated security evaluation model development focus on evaluating the attacks against the route discovery phase, we note the insider’s ability to drop packets at any time to highlight a significant security vulnerability that cannot be eliminated from two-phased routing protocols.

5.1.2 Classical Routing Attacks

There are several classical routing attacks against two-phased MANET routing protocols. These attacks include the Sybil attack [56], the invisible node attack [44,94], and routing wormholes [99].

In a Sybil attack, multiple malicious insiders share keying material and can operate as multiple identities during route discovery. If a Sybil attacker assumes, or spoofs, another identity during route discovery, the resulting route does not reflect the respective network topology. Since the Sybil attacker operates with multiple identities, the attacker is not bound to continue to perform as the forged identity during the subsequent data communication over the discovered route.
Additionally, malicious activity detection mechanisms attribute routing failures to the spoofed identity, not the malicious node.

The invisible node attack (INA) occurs when a node participates in a routing protocol without revealing its identity. An invisible node simply forwards messages during route discovery. Any discovered route passing through the invisible node reports a path that does not reflect the actual network topology.

Routing wormholes utilize two nodes to create a tunnel or special out-of-band network to either make a route appear shorter than it is or to completely hide one wormhole endpoint.

Although there have been numerous attempts to resolve these attacks [35,44,94,99,100,101], no solution has yet to provide a guaranteed defense [97,102]. The core element enabling these attacks is the inability to positively identify ones' neighbors. It may be tempting to pass these attacks onto a different mechanism, such as network discovery. However, the outcome is seen as an attack against the routing layer. Since these attacks cannot be eliminated, they must not be ignored during secure routing protocol analysis. For instance, the authors in [21,45,54] incorrectly claim that the Sybil and INA attacks have been eliminated and claim their developed protocol is provably secure.

One may also inappropriately maintain that if an attacker delivers a service during route discovery, then the attacker continues to provide that service. For instance, if node A claims to be node B during route discovery then all discovered paths that report B in place of A continue to be serviced by node A. However, node A has no obligation to forward subsequent packets addressed to flow through node B and the routing protocol attributes any failures to node B. Once a route fails, the protocol is forced to use another route or initialize a new route discovery process.

In addition to classical attacks, we must also determine if protocol message construction allows an attacker to corrupt the route discovery process. That is, we must search for an attacker's ability to utilize information revealed by protocol messages to actively corrupt the embedded route in on-demand source routing protocols. We develop the adaptive threat model and subsequent automated security evaluation models to search for these additional route corruption attempts and to identify the attacker capabilities under which a protocol may fail.

5.2 Canonical Threat Modeling

Modeling attacker capabilities presents a considerable challenge to accurately evaluate MANET protocol security. In the security protocol community, the Dolev-Yao [6] model delivers the strongest formal model to effectively evaluate authentication or cryptographic protocols. In the secure routing community, the attacker model rarely follows formalized attacker models.
5.2.1 The Classical Dolev-Yao Attacker Model

The Dolev-Yao model is the established approach to formally model attackers against authentication protocols. Dolev and Yao define the attacker as: “someone who first taps the communication line to obtain messages and then tries everything he can to discover the [shared secret]” [6]. They additionally establish the following attacker characteristics:

1. The attacker hears all messages.
2. The attacker is a trusted user and can initiate a connection to any node.
3. The attacker can be the connection target for any node.

The Dolev-Yao attacker model also assumes perfect cryptography. That is, it assumes all cryptographic primitives are perfectly secure and brute force attacks against keys enabling these mechanisms cannot be executed in polynomial time.

During analysis, the Dolev-Yao attacker exploits information obtained from captured messages to replay, modify, or create new messages in order to gain unauthorized secret information. Formal analysis techniques to evaluate authentication protocols frequently model the initiator and target nodes as endpoints and the Dolev-Yao attacker as either a tap into the communication channel or as a central entity through which all communication is channeled [66], as illustrated in Figure 16.

![Figure 16. Classical Dolev-Yao Attacker](image)

Figure 16a follows the original Dolev-Yao specification by tapping the media. The network cloud indicates that the intermediate nodes comprising the physical path are irrelevant to the end-to-end security requirement for the authentication protocol. The attacker can capture any message and can inject a message at any time. The attacker can replace any message, since message deliverability between an initiator-target pair cannot be guaranteed in a distributed environment. That is, dropped messages between an initiator-target pair are modeled by allowing the attacker the ability to replace messages.

While Figure 16a more realistically reflects real-world implementations, modeling the attacker as a central entity, illustrated in Figure 16b, provides equivalent security analysis capabilities. Following the central entity approach, the attacker is part of the communication channel between the initiator and target. The modeled attacker simply relays all messages between the initiator-target pair until it accumulates the information required to inject messages to break the protocol. When evaluating authentication protocols with a centralized attacker, one must remind
themselves they are not modeling the physical communication hops, but modeling an end-to-end message abstraction.

In either case, the attacker hears all messages. Since the attacker captures all messages, it can build its knowledge base by extracting information, such as a session key or nonce\(^9\), which may enable the attacker the ability to create, modify, or replay packets in order to break an authentication protocol. Formal and automated approaches typically follow the later approach represented by Figure 16b, modeling the attacker as a central entity or as the media itself. The centralized approach is easier to formally specify and reduces code complexity when implementing automated model checking evaluations.

5.2.2 MANET Attacker Models

While the Dolev-Yao approach provides an effective means to formally model attacks against authentication protocols, modeling attacks against MANET routing protocols poses a different requirement.

The goal in an **authentication, or cryptographic, protocol** is generally to share a secret between an authenticated initiator-target pair. The Dolev-Yao approach can effectively encapsulate attacks against an authentication protocol’s end-to-end security requirement. As Figure 16 indicates, the end-to-end security requirement does not consider intermediate nodes within the communication path. Authentication protocol security evaluations usually do not consider attacks against the path between the initiator-target pair.

Conversely, a MANET **secure routing protocol** must ensure the route discovery process delivers routes that reflect the network topology. Ensuring path integrity and subsequently using intermediate network hops to deliver a message between an initiator-target pair is the primary objective for the secure routing protocol. Thus, the actions taken by the intermediate nodes are significant and must be included in the formal model during security evaluations over the route discovery process. In the context of secure ad hoc routing protocols, the Dolev-Yao attacker can be viewed according to Figure 17, where the attacker node A has a direct link to all network nodes.

![Figure 17. Dolev-Yao MANET Attacker](image)

\(^9\) A nonce is a random number used once in an authentication protocol instance.
The Dolev-Yao MANET attacker can capture any message in the network and can transmit a message to any network node. Since the attacker effectively reduces communication between any two nodes to a two-hop network channeled through the attacker, the Dolev-Yao attacker offers the strongest attacker model for evaluating MANET routing protocols. If a routing protocol is shown secure against a Dolev-Yao attacker, the protocol is secure against any polynomial bounded attacker capability.

Unfortunately, the Dolev-Yao model is not traditionally used to evaluate MANET routing protocols. The most common approach to model attacker capabilities used throughout the MANET community is to assume the attacker node has the same communicational capabilities as any node within the network. Forcing an attacker to use nodes without any additional capability unrealistically limits the attacker. Results from a limited attacker evaluation may produce security claims that can be subverted by eliminating the attacker restrictions. Fortunately, there have been some recent efforts to more formally model the attacker.

**Active-n-m attacker.** Work in [24] introduces the active-n-m formalized attacker model, where \( n \) is the number of compromised insiders holding keying material, and \( m \) is the total number of attacker nodes in the network. All attacker nodes in the active-n-m model have the same communication capabilities as non-malicious nodes, plus the nodes have the ability to distribute compromised keys to all other \( m-1 \) attacker nodes.

The authors in [45] use the active-n-m approach with an additional configuration limitation. They combine all neighboring attackers that share a direct link into a single node. The combined single attacker is consequently limited in its transmission capability from a single node location, changing the modeled network topology as illustrated in Figure 18. Forcing two neighboring attackers to be represented as a single entity restricts attacker capabilities by forcing attacker nodes to relay all messages received from any local attackers. The authors claim that the combined attacker view provides the new perceived network connectivity. However, as far as the other nodes in the network are concerned, their view of the given network topology follows protocol rules and assumptions on node capabilities. For instance, node \( A_2 \) in the real topology sends messages that node \( n_1 \) does not directly receive. Following the authors attacker combination view, \( A_2 \)'s message reaches \( n_1 \), since node \( A_{1,2} \) has a link to node \( n_1 \). This viewpoint forces network connectivity that does not exist, resulting in an incorrect analysis.

![Figure 18. Combining Local Attackers](image)
Para metric attacker. The parametric attacker approach in [68] further refines the active-n-m model. The parametric attacker, represented by \( A(k, S_{A}) \), specifies the number of attacker nodes \( k \) and the initial pre-distributed attacker knowledge \( S_{A} \), such as keys. The Dolev-Yao attacker is addressed as a special boundary case, in which each link includes a parametric attacker. Unfortunately, the authors do not indicate how the Dolev-Yao boundary case interacts with the protocol. Additionally, the authors do not permit colluding attackers to share information extracted from captured messages. The scenarios they evaluate include a single adversary with the same communication capabilities exhibited by the non-malicious nodes.

Both the active-n-m and the non-boundary case parametric attacker are scenario dependent, based on a attacker with the same capabilities as the non-malicious nodes. The analysis using these models selects a few isolated network configurations and does not consider altering the attacker’s capabilities [45,68]. These models cannot infer protocol security against attackers that are not bound to follow their modeled restrictions or in network scenarios that were not analyzed. Regardless of their limitations, these attacker models are incorrectly used to claim protocol security.

5.3 An Adaptive Threat Model

We wish to maintain Dolev-Yao attacker strength in the adaptive threat model in order to determine if the route discovery process can be violated. That is, we want to know if a route can be returned that is not consistent with the current network topology. It is imperative to focus on possible route violations as opposed to probable attacks based on network configurations, number of attackers, or attacker strength, since our goal is to determine if routing attacks exist against a given protocol, regardless of the environment. In this modeling approach, if a protocol is secure against the Dolev-Yao attacker it provides security against all polynomial bounded attackers.

Although the Dolev-Yao attacker is the strongest modeling approach, modeling the strongest attacker does not offer the precision to identify the minimum capabilities necessary to break a protocol. Understanding the minimum capabilities required to break a routing protocol allows us to determine the environments in which a protocol can successfully find trusted routes.

Assuming an attacker cannot break cryptographic primitives in polynomial time, the attacker capability, or strength, is determined by:

1. The attacker’s communication range.
2. Whether the attacker is an insider or an outsider.
3. Whether a single or multiple attackers exist.

An attacker's communication range and ability to share information with other attackers relate to the attacker’s capacity to learn information required to break a protocol. The attacker’s status as an insider vs. an outsider determines the messages the attacker can generate.
By developing an adaptive threat model, we attempt to determine the minimum attacker strengths required to corrupt routes returned by the route discovery phase. This approach follows work in [103] to look beyond Dolev-Yao capabilities during authentication protocol analysis, allowing the security analyst to evaluate different attacker environments in ubiquitous systems. Adapting the attacker capabilities enables us to identify the conditions in which an attack is possible and this model does not suffer from restrictions imposed by bounding the attacker to a single capability. At the same time, including the Dolev-Yao attacker guarantees that attacks that may have been missed by a weaker attacker due to network configurations are discovered.

5.3.1 The Model

We provide the adaptive threat model through the attacker classification shown in Table 5, tailored specifically to search for route integrity attacks against MANET routing protocols. Route integrity attacks corrupt the route discovery process, resulting in returned paths that do not exist for the given network topology. Analysis using the adaptive attacker views the mobile network as a snapshot in time, isolating failures to malicious activity. Using an adaptive attacker classification provides the ability to identify capabilities required to break a routing protocol. Evaluating a protocol against the spectrum of attacker capabilities results in a more complete security analysis outcome, rather than claiming security based on a restricted attacker. Security claims based on a single attacker capability may be easily subverted by an unlimited adversary or under different operational scenarios.

<table>
<thead>
<tr>
<th>Attacker Strength</th>
<th>Communication Capability</th>
<th>Insider/Outsider</th>
<th>Attacker Category</th>
<th>Attacker Goal on Route Integrity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Intruder</td>
<td>Same as non-malicious node</td>
<td>Outsider</td>
<td>I</td>
<td>Add self to route, Corrupt route</td>
</tr>
<tr>
<td></td>
<td>Unlfinite receive radius</td>
<td>Outsider</td>
<td>III</td>
<td>Add self to route, Corrupt route</td>
</tr>
<tr>
<td></td>
<td>Transmission radius same as non-malicious node</td>
<td>Outsider</td>
<td>IV</td>
<td>Corrupt route</td>
</tr>
<tr>
<td></td>
<td>No limitations (Dolev-Yao)</td>
<td>Outsider</td>
<td>V</td>
<td>Add self to route, Corrupt route</td>
</tr>
<tr>
<td></td>
<td>Same as non-malicious node</td>
<td>Insider</td>
<td>VI</td>
<td>Corrupt route</td>
</tr>
<tr>
<td>Multiple Intruders (all intruder keys shared)</td>
<td>Same as non-malicious node</td>
<td>Insider</td>
<td>VII</td>
<td>Corrupt route</td>
</tr>
<tr>
<td></td>
<td>Unlimited receive radius</td>
<td>Insider</td>
<td>VIII</td>
<td>Corrupt route</td>
</tr>
<tr>
<td></td>
<td>Transmission radius same as non-malicious node</td>
<td>Insider</td>
<td>IX</td>
<td>Corrupt route</td>
</tr>
</tbody>
</table>

Following the attacker classification, an outsider node can capture any messages transmitted within its reception range, can replay captured messages, and can generate messages from information it has recovered from original knowledge or captured messages. The attacker’s goal against MANET routing protocols is to return a route that does not exist in the current network topology.
The specific effects depend on whether the attacker is an outsider or a trusted insider. The malicious outsider’s goals are to either corrupt the route so that an invalid path from the source to the destination exists or to add itself to the route, since it is not an authorized node. Insider nodes have the additional ability to sign messages, as they hold trusted cryptographic keys. Since malicious insiders are authorized users, adding themselves to a valid route does not constitute an attack for the malicious insider. Therefore, malicious insiders only attempt to corrupt the route.

In order to determine the required capabilities to attack a protocol, the adaptive attacker classification examines both single intruders and multiple intruders, along with various communication capabilities an attacker may have. The attacker capabilities range from having the same communication capabilities as a standard node in the network to having no transmission or reception limitations by using a Dolev-Yao attacker. The canonical Dolev-Yao model assumes the attacker is a trusted insider. We further refine the Dolev-Yao case by allowing the attacker to be an outsider or an insider. This refinement enables us to determine if an attacker without communication limitations has different effects based on whether the adversary has trusted keys.

Between the standard node attacker and the Dolev-Yao extremes, we add an asymmetrical attacker with an unlimited reception capability and a limited transmission capability. This category (attacker classifications III, IV, and VIII) does not assume the attacker node follows bidirectional communication rules, allowing the attacker to appear as a normal node during transmissions, while at the same time allowing any message to be received. An attacker with this capability can arise by having a more powerful transceiver or antenna than the standard network nodes, where the attacker can restrict its transmission range by adjusting its output power.

We illustrate the asymmetrical attacker using Figure 19. The attacker can capture all network messages, allowing it to craft routing attacks if it obtains enabling information. Because the attacker restricts its transmission to nodes S and D, nodes n₁ and n₂ are unaware that a forward link exists to node A. If node A extracts the appropriate information from the network messages, the attacker may be able to remove node n₁ from the path S-n₁-n₂-D, resulting in the corrupt path S-n₂-D. We provide a detailed example attack in Section 5.3.2.

![Figure 19. Asymmetrical Attacker](image)

We also use unlimited reception range to eliminate any reliance on special network topology configurations that may help enable an attack. By allowing an attacker to capture all network traffic, we inherently consider any network topology, attacker position, or additional capabilities...
provided by collusion between individual attacker nodes. Therefore, the adaptive attacker model preserves all capabilities of the active-$n$-$m$ and parametric approaches. For instance, work in [21,45] uses the active-$n$-$m$ approach to report numerous attacks against the Ariadne [24] protocol. These attacks are based on various network topologies, including some instances requiring colluding nodes. The authors' various scenarios allow the attacker to hear information that it can use to generate a corrupted routing message. The adaptive threat model's unlimited receive attacker category (categories III, IV, and VIII) does not require crafty scenarios to receive the correct enabling information, since the attacker inherently captures all publicly transmitted messages.

The adaptive threat framework also allows us to model multiple intruders. For multiple intruders, we assume the attackers are always colluding insiders. Multiple outsider attackers are already implicitly modeled by the single no-limitation outsider category (attacker classification V). Since malicious outsiders do not have cryptographic keys to share, colluding outsiders simply share information. Information shared from different portions of the network is already modeled by the category V attacker.

We also assume that all malicious insiders collude, since multiple attackers not working together do not have any greater individual capability to corrupt the route discovery process than a single attacker. If the attackers follow the common communication capability (attacker classification VII) and are not within each other’s transmission range, we permit the colluding nodes to at least share cryptographic keys. Key sharing between colluding nodes allows us to seek attacks enabled by the ability for a single node to sign or decrypt information computed with the colluding attacker's key.

If we allow an unlimited receive radius in a multiple colluding attacker environment, nodes in essence share captured information by receiving the same messages obtained during protocol operation, regardless of the network configuration. Sharing information through this mechanism has the same effect as the multiple colluders setting up a secret out-of-band communication mechanism or having non-advertised intermediate colluding nodes that simply relay information between two malicious attackers.

The adaptive threat model additionally allows us to inherently model attacks based on event ordering, or messages timing, between intermediate nodes. As an example of a timing based attack, one attacker scenario in [45] is based on an active-2-2 configuration and specifies message delivery order. The authors ensure that intermediate nodes receive crafted route replies to bypass sequence number checks in order to transmit the required corrupted messages. Sequence numbers are commonly used to guard against message replay between route discovery rounds\textsuperscript{10}. Sequence numbers are also used to reduce message overhead in one route discovery round.\textsuperscript{10}

\textsuperscript{10} A route discovery round is a complete route discovery process initiated by one source to find a path to the source's desired destination.
protocol round to ensure intermediate nodes respond to a given route request only once during the given round.

Instead of crafting a scenario based on the expected message timing, the adaptive threat model’s no-limitation categories V, VI, and IX inherently allow all possible message sequences. We base the approach on the fact that packet deliverability is not guaranteed in a wireless networking environment. The authors in [45] assume that if a node already processed a route request that any further route request messages from the attacker are ignored. Since the messages the authors rely on may be lost in the wireless environment, we must allow the possibility for any node to interact during the route discovery. That is, if it is possible for an attacker node to break the route discovery process, regardless if a non-malicious node has already processed a message for that sequence number along a shorter route, the routing protocol is not secure due to the fact that we cannot ensure another route has already been received. Again, we are evaluating the routing protocol for possible attacks, not probable attacks.

5.3.2 An Example

Consider an example using the adaptive attacker classification for a single malicious insider (attacker classifications II, IV and VI). We evaluate signature-based Ariadne [24] against the topology in Figure 20.

![Figure 20. Single Attacker Example](image)

During the route request (rreq), each intermediate node appends its identification to the route path, computes a hash value, generates a signature over the new packet, and retransmits the rreq containing the new path, new hash value, and appended signature. For example, msg₂ is constructed as:

\[ \text{msg}_2 = (\text{rreq}, S, D, h_{n1}, (n_1)), (\text{sig}_{n1}) \]

where the hash value \( h_{n1} \) results from the one-way hash function \( H \) according to the function \( h_{n1} = H(n_1, h_s) \). The value \( h_s \) was included in \( \text{msg}_1 \) and based on a shared secret between \( S \) and \( D \).

In the example scenario of Figure 20, node \( S \) attempts to discover a route to node \( D \), with node \( A \) being a malicious insider. The achievable routes according the topology are \( S-A-D \), \( S-A-n_2-D \), \( S-n_1-n_2-D \), and \( S-n_1-n_2-A-D \). We recognize \( A \)'s ability to drop packets once it is included in a
route and $A$’s ability to perform an invisible node attack, since we have already established these attacks exist. Here, we are searching for vulnerabilities enabled by information revealed within the protocol messages. Therefore, we are searching for $A$’s ability to trick the protocol into accepting an invalid path consisting of $S-n_1-A-D$ or $S-n_2-A-D$.

Using the attacker category II, the adversary has the same communication characteristics as all other nodes in the network. When $A$ receives $msg_3$ from $n_2$, it removes $n_2$ from the path and drops $n_2$’s signature. The per-hop, one-way hash value $h_{n_2}$ is intended to guard against $A$ taking this action. However, $A$ can generate $n_1$’s hash embedded into $msg_2$ directly from $msg_1$, since $h_{n_1} = H(n_1, h_s)$ and $n_1$’s identification is public. The destination node $D$ validates the hash value and signatures, accepting the path as $S-n_1-A-D$, but notice that the path $S-n_1-A-D$ is not valid according to the network topology. Node $D$ then creates a signed route reply ($rrep$) in $msg_5$, through $A$ to node $S$. The complete message transmission for the attack is shown in Figure 21, where $*msg_4$ introduces the malicious path.

\[
\begin{align*}
msg_1 &= (rreq, S, D, h_s()), (\text{sig}_s) \\
msg_2 &= (rreq, S, D, h_{n_1}, (n_1)), (\text{sig}_s, \text{sig}_{n_1}) \\
msg_3 &= (rreq, S, D, h_{n_2}, (n_1, n_2)), (\text{sig}_s, \text{sig}_{n_1}, \text{sig}_{n_2}) \\
*msg_4 &= (rreq, S, D, h_A, (n_1, A)), (\text{sig}_s, \text{sig}_{n_1}, \text{sig}_A) \\
\text{where } h_A &= H(A, h_{n_1}) = H(A, H(n_1, h_s)) \\
msg_5 &= (rrep, S, D, (n_1, A)), (\text{sig}_D).
\end{align*}
\]

**Figure 21. Example Attack Sequence**

This attack mirrors the attack presented in [21], showing how the adaptive attacker model captures the active-$n$-$m$ attacker. Notice that the attack depends on the network topology. In order for this attack to occur, node $A$ must be able to receive $msg_1$, $msg_3$, and $msg_5$, and be able to transmit $msg_4$ to $D$ and relay $msg_5$ to $S$. Consider that if the link between $n_2$ and $A$ did not exist, $A$ would not recover $\text{sig}_{n_1}$ from $msg_2$. We can produce the same result if we allow $A$ the ability to receive all messages. If $A$ has an unlimited receive capability (attacker category IV), it can extract $n_1$’s signature and hash value directly from $msg_2$, resulting in the ability to construct the same corrupted path $S-n_1-A-D$. In this scenario we still require the ability to send messages to $S$ and $D$.

The final classification for the single malicious insider (attacker category VI) follows the full Dolev-Yao model, which imposes no restrictions on topology requirements to duplicate the attack sequence in Figure 21. The Dolev-Yao attacker acts as a fail-safe in case the previous categories failed to identify an attacker due to the evaluated network topology. However, using only the Dolev-Yao model does not allowed us to identify the protocol's vulnerability to weaker attackers.

As the above example indicates, the category II attacker utilizing standard node communication capabilities relies on the chosen network topology. To ensure this attack is found
during evaluation against a category II attacker, all network configurations must be considered. Topology dependent attacks challenge the security analyst to a grueling manual evaluation process in order to evaluate all possible network configurations.

An alternative to manual exhaustive analysis is to utilize exhaustive automated analysis techniques such as model checking. We develop and present a model checking-based automated analysis technique in Chapter 6.

5.3.3 Classifying Current MANET Attacks

Section 5.3.2 illustrates how the adaptive threat model captures attacks that actively corrupt the MANET route discovery phase by extracting information from captured messages. Using the adaptive threat model in this manner can discover all similar attacks against the Secure Routing Protocol (SRP) and Ariadne presented in [21,45]. Our approach is not reliant on network topology to discover if a protocol is vulnerable to an unrestricted attacker. In this section we examine attacker categories within the adaptive threat model to determine if they capture the classical routing attacks discussed in Section 5.1.2.

The INA [44,94] occurs when a single node forwards routing messages without adding itself to the routing path. The attack can be intuitively described as a relay attack and provides the same capability regardless of whether the attacker is an outsider or an insider. For example, if node A in Figure 20 performed the INA, the corrupt route S-D is returned to the source S, tricking S and D into believing they are one-hop neighbors. The INA can be performed by any single-node attacker (attacker categories I – VI) regardless of network topology, noting that the weakest attacker (category I) has the capabilities required to perform the INA.

The Sybil [56] attack occurs when an attacker node assumes more than one identity. In the context of a secure routing protocol where signatures are used to enforce node identity, the Sybil attacker requires the knowledge of a colluding attacker’s key, since the attacker assumes multiple identities by signing packets with more than one key. The attacker does not interact or require any interaction with a colluding node once the key material has been obtained. The Sybil attack can be realized by any multiple-node attacker (attacker categories VII – IX), noting that the weakest multiple intruder attacker (category VII) can implement the attack. In essence, once colluding keying material is obtained, the attack is performed by a single node.

Finally, wormhole attacks [99] occur when two nodes set up a specialized channel to secretly tunnel messages between different parts of a network to force a route to appear shorter than it is. The tunneled channel is not part of the network topology and is therefore considered a corrupted path. To implement the specialized channel, the nodes making up the wormhole must be able to communicate with one another, regardless of the network topology. In the adaptive threat model, the out-of-band channel can be achieved by the multiple attacker classifications for category VIII and IX. Lifting the restrictions on the attacker’s reception radius allows the two worm-hole
endpoints the ability to communicate with one another. If the wormhole attacker chooses to hide
the identity of both wormhole endpoints, the effect can be captured by an unlimited single-intruder
Dolev-Yao representation (categories V and VI). The Dolev-Yao's unlimited single attacker
capability to capture any message in a network and transmit that message to any other location in
the network allows it to represent a wormhole attempting to implement an adapted INA.

5.4 Chapter Summary

In this chapter we meet one of the thesis objectives by developing an adaptive threat modeling
approach. An adaptive threat model provides a more comprehensive approach to modeling
threats against MANET routing protocols, advancing the ability to model the threat without limiting
an analysis based on attacker capabilities or network topologies.

The adaptive threat model supplies a foundational piece required for automated model
checking MANET routing security. The threat model’s relationship to developing the automated
model checking capabilities, presented in Chapter 6, is cyclic in nature. The need for the adaptive
threat model is driven by the automated analysis goal to provide exhaustive security analysis. On
the other hand, the numerous attacker scenarios provided by the threat model require automated
procedures to be effective and efficient.

Section 5.1 provided information on attack sources and included a discussion on classical
routing attacks. In Section 5.2 we discussed the various attacker threat models currently used to
evaluate the MANET route discovery process. We saw how the current threat models may overly
restrict the attacker's capabilities, resulting in claimed secure routing protocols that may be easily
attacked. We cannot assume an attacker follows the restrictions assumed by the analyst. Section
5.3 contains our primary contribution in the area by presenting the adaptive threat model for
MANET secure routing evaluations. Instead of claiming protocol security based on attacker
assumptions, we adapt the attacker capabilities in order to determine at what point a protocol
may fail. By adjusting the attacker communication capabilities, we do not rely on special network
topologies to enable attacks.

While we have illustrated a small handful of possible ad hoc routing attacks, we contend that
any attack against the ad hoc route discovery phase could be represented and discovered
utilizing the adaptive attacker classification. The adaptive model ensures attackers are discovered
via the unrestricted Dolev-Yao attacker, while at the same time provides the precision to
investigate minimum capabilities required to corrupt MANET routing protocols.
CHAPTER 6

AUTOMATED SECURITY ANALYSIS

In this chapter we develop a formal methods approach, following the model checking paradigm, to evaluate route validity for routes returned in on-demand source routing protocols.

Model checking [59] involves an automated process whereby a finite state model is generated and exhaustively searched to determine if the given security property holds. If the system fails, model checking can provide an event sequence that leads to the failure, thus automatically finding protocol security vulnerabilities. Automated formal methods have shown success in evaluating authentication protocols [60,65,69,92]. There has also been some initial work [67,93] using automated formal methods to evaluate MANET route security for ad hoc on-demand distance vector protocols such as AODV and Secure AODV (SAODV), where the attacker's goal is to corrupt a node's next-hop entry for destinations stored in the node's routing table. However, the work in [67,93] is isolated to evaluating a few network topologies. Evaluations that do not search the complete network topology space overlook attack possibilities.

Figure 22 illustrates our automated security analysis process that exhaustively checks all routing combinations to evaluate if an attacker can corrupt the route discovery phase by returning validated routes that are not consistent within the network topology. This process allows us to generate and examine all possible network topologies for a given sized network.

There are numerous model checking environments designed to analyze distributed protocols. The Failures-Divergences Refinement (FDR) model checker [66], SPANNER [104,105], and the SPIN model checker [63] are just a few. FDR uses the Communicating Sequential Processes (CSP) [106] formal language to exhaustively search all execution traces against the possible traces determined by the specification. SPANNER is based on the selection/resolution model, providing the ability to evaluate combined finite state machine representations to determine if a distributed system has unreachable states, deadlocks, or unwanted loop conditions. We use the SPIN model checker for our research based on the discussion in Section 6.1.

We transform the routing protocol, attacker actions, and security property into a model for analysis using the SPIN model checker [63,107]. The abstraction contains only the elements required to determine if the MANET route discovery mechanism can be corrupted by an active attacker. The topology generation engine, evaluation engine, and reporting engine provide exhaustive topology generation and enable SPIN execution and failure reporting over each possible network topology for a given sized network.
The remainder of this chapter is outlined as follows. Section 6.1 describes the SPIN model checker and its desirable features which make it attractive for modeling security properties in MANET routing protocols. Section 6.2 contains SPIN model development over the wireless medium, abstracting the wireless broadcast process required to model wireless communication. Section 6.3 describes SPIN model development for the source routing protocols DSR, SRP, Ariadne, and endairA. Section 6.5 provides SPIN attacker model development. Section 6.6 develops procedures to enable exhaustive automated topology generation, model execution, and failure reporting.

### 6.1 The SPIN Model Checker

We transform the routing protocol, attacker actions, and the desired security properties into a formal model for analysis using the SPIN model checker. Holzmann developed SPIN [63,107] to verify correctness properties for concurrent, or distributed, systems. To use SPIN, a protocol and its desired goals are specified in the Promela (Process Meta Language) formal modeling language. Promela was developed to facilitate forming a proper abstraction, which is essential to model checking completeness. Excessive modeling detail can lead to state-space explosion, rendering model checking unusable if the system states exceed the available memory. SPIN generates a finite state automaton (FSA) and verifies the claimed goal through exhaustive reachability analysis. If the protocol analysis encounters a failure according to the specified criteria, a counter-example is produced showing the event sequence leading to the discovered failure.

One of the factors in choosing SPIN is its success in verifying security properties in authentication protocols [108] and it has been used to evaluate the loop freedom criteria for
MANET routing protocols in non-malicious environments [95,109,110]. Our research combines these two areas to analyze security properties in MANET routing protocols.

Additional factors in choosing SPIN are its ability to model message passing, non-deterministic choices, and independent processes. These factors allow us to more precisely model the routing process in the wireless MANET environment.

SPIN models message passing following the approach used in CSP [106]. Message channels are formed by specifying the channel size and the associated message format. Nodes transmit and receive messages as follows, where ! indicates message transmission and ? indicates message reception:

- **Transmit**: channel_name ! message
- **Receive**: channel_name ? message.

Promela models non-deterministic choices based on Dijkstra’s guarded statement syntax [111], where :: indicates an entry, or guard condition, and -> indicates the operations executed if the entry condition is true. Promela's ability to code non-deterministic choices allows us to model the wireless message deliverability environment. Since wireless message deliverability is not guaranteed, an attacker may be able to inject false routing information in place of dropped messages. We cannot assume messages are delivered in wireless environments and must exhaustively analyze all possible message sequences to determine if attacks are possible.

Non-deterministic choices are modeled in SPIN using Promela's conditional statements and looping constructs. Unlike C, Promela conditionals and loop criteria can have multiple true statements. We can think of Promela's if construct, blocked within an if-fi body, as being similar to C's switch statement with more than one case being true. We use the if construct in Figure 23 to model wireless message reception, where a message may be non-deterministically processed or dropped. Since both entry statements are true, either statement and its corresponding actions may be non-deterministically chosen during SPIN simulations. In addition to simulating a single event sequence based on the initial seed value, SPIN can also be used to provide exhaustive analysis over all possible sequences. During SPIN exhaustive analysis, all possible non-deterministic choices are examined independently. Using SPIN in this fashion allows us to model all possible message interactions to determine if an attack may occur.

```
if
  :: message received ->
    process message according to protocol
  :: message received ->
    drop message and take no actions
fi
```

**Figure 23. Modeling Message Reception**
Promela’s ability to represent independent processes allows us to model the distributed MANET environment. Wireless MANET nodes operate asynchronously, with no common timing mechanisms. Each node operates independently, participating in the protocol process only when messages are received, choosing to respond in the routing process or disregarding the packet based on the protocol rules. For example, an intermediate DSR node that receives a req forwards that route request only if it has not previously processed the request, based on the initiator-target pair and the message identifier.

Once the protocol is modeled in Promela, SPIN generates a FSA. For example, where \( M \) is the system model for the protocol and its environment, \( S \) is the set of states, \( \rightarrow \) specifies the transition between states, and \( L \) is a labeling function that lists which values hold for a specific state, SPIN generates the FSA according to the form: \( M = (S, \rightarrow, L) \).

Prior to analysis, we specify the desired goal or correctness property \( (\phi) \). The desired property is stated as a liveness or a safety property. Liveness properties reflect the desired functionality the protocol must supply, such as a routing protocol must promptly return a route if one exists. Safety properties specify undesired properties that must be avoided, such as an attacker corrupting the returned route so that it is inconsistent with the network topology. According to Monin [112], liveness properties are used to indicate “something good happens,” while safety properties are used to ensure “nothing bad can happen.”

According to these definitions, we classify evaluating if route discovery can be corrupted as a safety property. That is, we never want the possibility to accept an invalid route. If a MANET routing protocol accepts an invalid route, it fails the safety constraint, resulting in a security failure.

Once the protocol model and desired properties are specified, formally stating that the system model \( M \) models the given security property \( \phi \) over all \( \sigma \) computational paths is captured by the following equation:

\[
(M \models \phi) \leftrightarrow \forall \sigma, (\sigma \in M \rightarrow \sigma \models \phi).
\] (1)

Since SPIN builds a FSA over the complete system, message timing is not explicitly modeled. State transitions in the system FSA do not solely model transmitted messages. State transitions also indicate internal processing at each node. To incorporate timing, SPIN views correctness specifications as a Linear Temporal Logic (LTL) property. LTL inherently models time through event sequences. To perform exhaustive automated analysis, SPIN searches for any event sequence that leads to a failure, indicating the message sequence leading to the attack. SPIN uses LTL to search for an attack possibility rather than looking at the attack probability based on the expected timing of messages between nodes. Discrete-event network simulators, such as OPNET, explicitly label all events occurring at the same time with the same time-step. This type of network simulator bases protocol security evaluations on expected message deliverability, not the possible event sequence. A security evaluation based on a probabilistic discrete event
network simulation does not exhaustively search all possible message event sequences, which does not prove a property exists or not. Using the SPIN model checker to exhaustively search all possible message sequences provides a sound security analysis approach to search if an attack is possible.

During exhaustive analysis, SPIN supports both depth-first-search (DFS) and breadth-first-search (BFS) algorithms. The BFS finds the shortest possible error sequence, while the DFS provides the first error encountered. The DFS algorithm is the default, since it is more efficient in SPIN to build the state-space on-the-fly as needed, to reduce the required state-space that must be generated before a failure is found.

As an example of a complete search space, Figure 24 illustrates the possible distributed process interleavings that must be evaluated for a simple two node implementation, with each concurrent node having two executable statements. In a real-world concurrent system, each node operates independently at the same time. In a SPIN model representation, only one statement executes at a time. To properly model the concurrency, SPIN non-deterministically determines which node exercises its next executable statement. SPIN’s exhaustive analysis capabilities search all event sequences between the concurrent nodes by building all possible combinations.

To illustrate a single case, let’s walk through the leftmost event sequence in Figure 24. After initialization we choose to either execute node 1 or node 2, choosing node 1 for the example. At this point, node 1 statement 1 executes (1.1), since statements inside a node execute in order. For the next execution, SPIN can decide to execute node 1 or node 2. Following the left path, we choose node 1 and execute that node’s remaining statement (1.2). For the next execution steps, SPIN has only node 2 to choose from and executes node 2’s statements in order (2.1, 2.2) for the next two steps. In total, we could have taken six different event sequences in the given scenario. By capturing all possible event sequences, SPIN is able to model the possible interactions between the concurrent nodes.

![Figure 24. Exhaustive State Space](image)

As the above example indicates, the state-space and possible message orderings rapidly grow with the number of concurrent processes being evaluated, as well as how many executable
statements exist in each process. SPIN uses various techniques to combat state-space explosion. The greatest effect on the required state-space, and most controllable by the security analyst, is to ensure that the protocol model precisely captures the elements required to evaluate the desired property. Even with proper model abstraction, system states can quickly grow beyond the memory available for exhaustive analysis.

SPIN increases its chances for exhaustive analysis by providing a state-space compression option that trades off space for execution time and can also reduce the state-space by using partial order reduction. The partial order reduction technique refers to a state-space elimination process that can occur if any concurrent process statement interleavings result in the same outcome, regardless of their execution order.

SPIN can also reduce the global state size through atomicity. If executable Promela statements are made atomic, then the complete atomic system block executes without being interrupted to interleave executable statements from other concurrent processes. For instance, if we make the statement sequences in node 1 and node 2 atomic in the preceding event sequence example, there are only two possible sequences: 1.1 → 1.2 → 2.1 → 2.2 or 2.1 → 2.2 → 1.1 → 1.2.

If exhaustive analysis is not possible with the state-reduction techniques, SPIN can utilize a bit-state hashing method to reduce the required storage space for each state [113]. On-the-fly generated states are normally stored to ensure that previously visited states are not re-examined. Bit-state hashes reduce the states via two independent hash functions. Future states are compared against their hash value to determine if they were previously analyzed. Since multiple states can hash to the same value, bit-state hashing cannot guarantee that each state has been previously visited. Since bit-state hashing does not provide 100% state coverage, a system evaluation with this method that does not identify errors does not guarantee that a failure does not exist. However, any failure identified via bit-state hashing provides a correct error sequence. In any case, exhaustive analysis should be performed if possible.

### 6.2 Modeling the Wireless Medium

As illustrated in Figure 25, we model three primary processes: the wireless medium, the non-malicious node, and the attacker node. We model wireless connectivity using a global connectivity array, which specifies the current network topology for the given evaluation. In SPIN, each modeled node instantiates its respective process type. For instance, a network model with five non-malicious nodes and one attacker initiates five instances of the non-malicious node, one attacker instance, and one wireless medium instance. While the wireless medium in itself is not a node, it directs all traffic for the wireless model.

One of the first priorities in building a SPIN representation for wireless ad hoc networks is to accurately model message transmission. Modeling message transmission in a wireless environment is a significant challenge for simulating and model checking MANET protocols.
When a wireless node transmits a message, all nodes within its transmission footprint receive the communication. This type of communication is known as broadcast communication\(^{11}\). Each wireless recipient then determines whether to process or drop the received packet.

![Figure 25. MANET Model Abstraction](image)

In real-world wireless implementations, all nodes within the sender’s footprint receive broadcast transmissions virtually simultaneously. SPIN does not natively support broadcast communication. When modeling broadcast communication in a finite state model checker, a single FSA event occurs at a time. Therefore, we model the required communication components to ensure that all neighbor nodes receive appropriate transmissions.

In [114], Ruys’ discusses how to implement simple broadcast communication in SPIN using a common bus structure, a matrix of channels for each node combination, or as a broadcast server process. Ruys describes how each of these options can be implemented in SPIN along with their associated features and limitations.

In the bus structure approach, modeled nodes communicate via a single communication channel. In SPIN we can model the channel as a queue, where transmitting nodes add messages to the queue. Tuning the modeled channel, or queue, size is critical since no messages can be added to a full channel. If a channel is too small and becomes full, the transmitting node must either drop the packet or wait until the channel becomes free, which may lead to a deadlock. If we try to combat a full channel by making the channel large, we encounter computational memory issues by increasing each state’s storage requirements. During state-space analysis, the channel contents are stored in each state. In a bus structure all nodes must check the bus channel and determine if they are the message recipient during unicast communications. Unicast messages are used for message transmission in source routing protocols after route discovery. To use the bus approach to support the flooded broadcast messages used during route discovery, each node must check the modeled channel and also determine if they are within the transmitter’s

---

\(^{11}\) Broadcast communication to all wireless nodes within transmission range should not be confused with broadcast addressing where messages are intended for all reachable recipients.
footprint. Additionally, the last node checking the bus must remove the message from the channel queue to allow future transmissions to enter the modeled channel. However, the distributed independent wireless nodes have no way to ensure which nodes have already received the message and if the message channel needs to be polled by additional nodes.

In the channel matrix approach, a communication channel is modeled for each possible communication link. The transmitting node determines the message recipients and sends separate unicast transmissions to each node over the respective dedicated channels. Route discovery messages using flooding are sent to each node in the transmitter’s footprint. Once the route is discovered, the transmitting nodes directs its communication to the next hop’s designated channel. The drawback to the channel matrix approach is that for a network of size $N$, $N^2$ dedicated channels are required, which complicates the state-space and does not allow us to seamlessly expand communication modeling to large scale systems.

Ruys’ final approach uses a special broadcast server process to direct all communication. The broadcast server receives messages transmissions and directs the message to each recipient along a designated channel for each recipient. This approach requires $N+1$ channels. The extra channel represents an incoming many-to-one message channel to the broadcast server process, to which all nodes direct their communication. An additional SPIN optimization over the $N+1$ dedicated channels can utilize exclusive receive operations to further reduce the state-space.

We model MANET communication between wireless nodes using a wireless medium server, similar to Ruys’ broadcast server process. By separating the modeled wireless medium into its own process rather than modeling direct node-to-node communication channels or using a centralized variable length channel bus, we limit the state-space and modularize our approach to eliminate modeling dependencies based on the network size being evaluated. The wireless medium server controls the analysis model. It contains knowledge over the network topology and knows which nodes are malicious or non-malicious.

We model a node’s transmission range by connectivity rather than by distance, by ensuring only reachable neighbors receive transmitted messages. We map the static network topology for the time instance we are evaluating into a two-dimensional array. Figure 26 illustrates the associated array for a four node network topology. The modeled wireless medium process uses the array to determine the transmitting node's local neighbors and sends the messages accordingly. The shaded areas indicate array symmetry if analyzing a bidirectional, or symmetric, environment. The array is composed of $N$ rows by $N$ columns, where $N$ is the total number of non-malicious nodes plus the total number of malicious nodes. Rows indicate the transmitting node and columns indicate the node’s local neighbors. Each array element holds a Boolean value, with true (or 1) indicating a communication link exists between node pairs.
Combining the independent wireless medium process with the global connectivity array allows us to model any MANET routing protocol. However, we tune the wireless medium server to model on-demand source routing protocols in order to maintain a reduced state-space. Figure 27 provides a high level view of the wireless medium server that captures the wireless communication process.

Listen for messages
- If rreq
  - Transmit unicast message to all the sender's local neighbors
- Else (is a rrep)
  - Transmit message to next upstream node in embedded path
  - Transmit message to any attacker nodes in the sender's transmission footprint

Go back to “Listen for messages”

To model the route discovery process, the wireless medium server delivers the forward broadcast rreq messages to each adjacent node. Each recipient node processes the message and responds with a subsequent message to the wireless medium server. The unicast rrep can use an identical process, with the exception that each node receiving the message must determine if it is included in the path before retransmitting the message to the next upstream host. However, in order to reduce the state-space, we model the wireless medium server to only transmit to the intended recipient identified in the unicast rrep message and to all attacker nodes within the current transmission footprint. Limiting the modeled messages reduces the state-space, yet captures the protocol's required elements for analyzing attacks against the route discovery phase in source routing protocols.

The resulting SPIN generated FSA for the modeled wireless medium server process, shown in Appendix A, contains 17 states. The total number of states is important when combined with the entire model to build an overall system FSA. The overall system FSA combines the wireless
medium with malicious and non-malicious nodes. We discuss the complete system state requirements further in Chapter 7.

6.3 Model Validation

The SPIN Promela models developed in the following section must be validated prior to their use in analyzing ad hoc routing security. Validation ensures the protocol model accurately captures the desired protocol features, which increases the confidence in the associated analysis outcomes. Jain defines model validation accordingly: “…if correctly implemented, the model would produce results close to that observed in the real system” [115]. Jain goes on to state that simulation models must be validated against at least one of the following three sources:

1. Expert intuition.
2. Real system measurement.
3. Theoretical results.

Expert intuition draws on peer interaction, such as dialogue with a protocol’s original designer or subjecting models to the peer review process received through the journal and conference submission process. Since Jain’s discussion focuses on performance modeling, validation against real system measurements entails statistical comparison between the real measurement and the simulated outcome. For instance, network simulation models developed to analyze performance are typically validated by comparing the statistical average over independent simulation runs against the performance metrics obtained from the physical protocol implementation [46,115]. In many instances in early research efforts, an actual system is not available for comparison against the model. In scenarios where physical implementations do not exist, network simulations can be analytically validated against the specification according to Jain’s third validation source.

In general, model checking paradigms are not intended to evaluate performance metric criteria. Instead, model checking’s goal is to prove a property holds or fails for a given scenario through exhaustive analysis. Prior to analysis, we must validate that the SPIN Promela models capture the desired protocol operations required for the analysis. Model checking paradigms can be validated against two of Jain’s criteria, namely validation against expert intuition or theoretical results. Using real system measurements for comparison is not applicable in model checking paradigms, as exhaustive analysis against a failure property is not a statistical performance criteria.

We validate the Promela protocol models against Jain’s third criteria, theoretical results. For each modeled protocol, we ensure the route paths are constructed according to the protocol descriptions and any protocol security mechanisms operate as desired. For instance, a protocol may be designed to use a message authentication code (MAC) to determine if the route reported to a target node has been corrupted during the $rep$ sent back to the initiator. To validate the
security mechanism, the protocol must detect any direct route corruptions we inject into the rrep since the MAC does not match the corrupted route.

6.4 Modeling Source Routing Protocols

One of the required objectives to meet the thesis goal is to automate the security analysis process in order to evaluate routing attacks. We focus on attacks against the route discovery phase in on-demand source routing protocols. On-demand source routing protocols are reactive in nature. They utilize a two-phased route discovery mechanism to establish a route to a given destination only when required. The selected route is used to transmit data during the data forwarding phase. Therefore, the route must exist within the network topology to be usable.

In order to model the protocols within the constraints of a finite model checker, we make several modeling choices to simplify the resulting model and limit the resulting state-space. Model checking over the simplified protocol representation exhaustively examines all routing possibilities, based on possible event sequences. Exhaustive analysis provides the ability to ensure an attack does or does not exist for a given scenario.

In the remainder of Section 6.4, we model the following source routing protocols: DSR, SRP, Ariadne, and endairA. DSR is a well known ad hoc source routing protocol. The additional protocols are security extensions based on the core DSR functionality. For all modeled protocols, we assume that the links between nodes are bidirectional, thus any target receiving a rreq returns the rrep over the accumulated path constructed during the rreq process. The bidirectionality assumption follows the evaluated protocol requirements and is not a side-effect, or requirement, of the modeling approach. Our modeling approach can easily model unidirectional, or asymmetric, protocols by setting the appropriate communication links in the model’s network connectivity array.

6.4.1 Modeling DSR

The DSR protocol is a well known reactive ad hoc on-demand source routing protocol [14]. Although DSR does not include any security goals, we build initial Promela DSR models to ensure that the SPIN finite space model checker can successfully model source routing protocols. The subsequently modeled protocols utilize the core DSR model development.

DSR model development. DSR constructs a route to a desired destination when the route is needed. Figure 3 on page 7 and Figure 28 below illustrate DSR route discovery by providing a high level description over the route discovery process.
**rreq process:**

- Initiator node
  - Initiate a *rreq* to target
- Intermediate nodes
  - If previously seen *rreq* → take no action
  - Else
    - If not target → append id to path and retransmit *rreq*
    - If target → generate *rrep*

**rrep process:**

- Target node
  - Unicast the *rrep*
- Intermediate nodes (along the unicast path)
  - If not initiator → retransmit *rrep*
  - If initiator → accept route

---

**Figure 28. High Level DSR Route Discovery Process**

The DSR route discovery message format can be viewed as:

<`msg_type`, `initiator`, `target`, `id`, `accum_path`>.

The `msg_type` labels the message as a *rreq* or a *rrep*, with the `initiator` and `target` indicating the source-destination pair for the desired route. The `id` is a unique identifier used to ensure a node forwards a given *rreq* once, the first time it sees the request. As many *rreqs* occur in an operational setting, each node tracks the `<`initiator, `id`>` for each *rreq* received. In the protocol model, we focus on a single route discovery process at a time. Therefore, we remove the `id` from the message and apply a simple Boolean flag in each node that tracks if the given node has sent a *rreq*. The accumulated path (`accum_path`) lists all intermediate nodes in the path and is updated at each node that processes the *rreq*. In the model, we list all nodes in the accumulated path, to include the initiator and target nodes. Listing all nodes allows us to subsequently use this field to check against the network connectivity array during analysis. We discuss the analysis check further in Chapter 7. We also add a position value (`accum_pos`) to track the array element that the current node adds its own `id` to. The modeled DSR message format follows as:

<`msg_type`, `initiator`, `target`, `accum_path`, `accum_pos`>.

Once the *rreq* is delivered to the intended target, a *rrep* is generated and sent back to the initiator. During the *rrep*, the accumulated path and the current array position are read by the wireless medium server to determine the next-hop destination node for the unicast *rrep*. As previously mentioned, wireless *rrep* messages in real-world physical implementations are received by all nodes within the current node’s transmission footprint and each receiving node determines if it is the intended next-hop in the explicitly labeled unicast transmission. While normally each receiving node decides whether to process and retransmit the unicast *rrep*, we
model the routing decision in the wireless medium process and send a single message to the next-hop identified by the embedded route. We also send the transmission to any malicious nodes that may be within the current transmission footprint. By modeling message transmission in this manner, we reduce the model state space by decreasing the number of modeled message transmissions.

The resulting SPIN generated FSA for the modeled DSR non-malicious node, shown in Appendix A, contains 21 states.

**DSR model validation.** Following the validation criteria formed in Section 6.3, we validate the DSR model to ensure routes are discovered according to the protocol specifications. We evaluate the SPIN produced messages for the 5-node network topology shown in Figure 29. In this configuration, node 0 is the initiator and node 4 is the target. The DSR Promela models are constructed such that node 0 attempts to establish a path to the highest labeled non-malicious node, which in this case is node 4. During validation, the target is allowed to respond to all received rreqs, resulting in the following possible routes: 0-4, 0-1-2-4, and 0-1-2-3-4. Since DSR does not implement any security mechanisms, no further model validation is required.

![Figure 29. Path Validation Topology](image)

Figure 30 contains SPIN generated output for the three returned paths at the initiator node 0.

---

Path 0-4:

```plaintext
from_wm_local?rrep,0,4,4,255,255,255,0
Route Discovered
```

Path 0-1-2-4:

```plaintext
from_wm_local?rrep,0,4,1,4,255,0
Route Discovered
```

Path 0-1-2-3-4:

```plaintext
from_wm_local?rrep,0,4,1,2,4,0
Route Discovered
```

---

**Figure 30. DSR Path Validation**

In the DSR validation output, the bold underlined portions in each message indicate the accumulated path constructed during the rreq, as it is returned to the initiator in the rep. The 255 values are unused portions of the accumulated path. The finite accumulated path array must be
large enough to accept the longest available path length \((N)\), in case all nodes in the network topology are included in the path. The from \(\text{wm\_local}\) prior to the receive operator \(\leftarrow\) indicates the communication channel. The \(rrep\) after the receive operator indicates the message type. The initial non-bolded \(0,4\) indicates the initiator-target pair. The trailing \(0\) during the \(rrep\) indicates the array position to identify the next-hop to which the wireless medium transmits the unicast message.

6.4.2 Modeling SRP

The Secure Routing Protocol (SRP) [23] provides an extension to DSR, attempting to secure the route discovery phase from maliciously corrupted routes. SRP assumes pre-existing security associations between any initiator-target node pair. Intermediate nodes do not utilize any cryptographic mechanisms during the route discovery process. Therefore, they interact following the DSR routing process.

The SRP message formats follow below. The bold underlined portions indicate the message components contained in the keyed MAC that is computed using the security association between the initiator-target pair and denoted as \(\text{MAC}_{it}\):

- \(<\text{rreq}, \text{initiator}, \text{target}, Q_{id}, Q_{sn}, \text{MAC}_{it}, \text{accum\_path}>\)
- \(<\text{rrep}, \text{initiator}, \text{target}, Q_{id}, Q_{sn}, \text{MAC}_{it}, \text{accum\_path}>\).

Figure 31 illustrates the SRP protocol messages for the given network topology, with initiator node 0 and target node 3. The query id \(Q_{id}\) and query sequence number \(Q_{sn}\) are used to ensure the route request is unique and has not been replayed. Similar to DSR, the intermediate nodes check the \(Q_{id}\) to ensure only the first \(rreq\) that is received is forwarded for the given \(<\text{initiator}, Q_{id}, Q_{sn}>\) value.

**SRP model development.** We base the SRP model development on the high level description over the SRP route discovery process in Figure 32 below, derived from the SRP message format and the SRP protocol from Figure 31.

The SRP route discovery process is similar to DSR, with changes added for the MAC generation and associated validation checks in the initiator and target nodes. Following the DSR modeling approach for a single route discovery attempt, we do not explicitly include the \(Q_{id}\) and \(Q_{sn}\). The intermediate nodes use a Boolean flag to determine if they have already forwarded the \(rreq\) and do not respond to more than one \(rreq\). The pairwise keyed \(\text{MAC}_{it}\) value during the
forward $rreq$ guards against replay attacks, given that the MAC computation includes unique $Q_{id}$ and $Q_{sn}$ values for a given route discovery attempt. Since we assume that polynomial time bounded attackers cannot break the cryptographic MAC, we do not explicitly model the MAC computation during the forward $rreq$ process. The MAC during the return $rrep$ is modeled to focus on possible attacks that corrupt the path discovered for an individual route discovery attempt. This approach reduces the modeled state-space.

We target the model development for attack analysis against the accumulated path ($accum\_path$), ensuring the SRP model can properly account for the target MAC over the accumulated path in the $rrep$. The modeled SRP message format follows as:

$<msg\_type, initiator, target, accum\_path, accum\_pos, mac\_path>$.

During the route discovery process, the target node models the MAC over the accumulated path by copying the path received in the $rreq$ into the $mac\_path$ variable and adding it to the $rrep$. The modeled MAC is protected against attacker corruption since it can be accessed only by the initiator-target pair due to their security association. We model the MAC's cryptographic property by not allowing intermediate nodes to access the $mac\_path$ value. Once the initiator node receives the $rrep$, it checks the accumulated path against the path contained in the $mac\_path$ value before accepting the route.

---

**$rreq$ process:**

- Initiator node
  - Compute MAC over initiator, target, $Q_{id}$, and $Q_{sn}$
  - Initiate $rreq$ to target

- Intermediate nodes
  - If previously seen $rreq$ → take no action
  - Else
    - If not target → append id to path and retransmit $rreq$
    - If target → take actions below in $rrep$

**$rrep$ process:**

- Target node
  - Validate MAC contained in $rreq$
  - Compute new MAC over initiator, target, $Q_{id}$, $Q_{sn}$, and accumulated path.
  - Unicast the $rrep$

- Intermediate nodes (along the unicast path)
  - If not initiator → retransmit $rrep$
  - If initiator
    - Validate MAC contained in $rrep$
    - Accept path if MAC validates

---

**Figure 32. High Level SRP Route Discovery Process**
The resulting SPIN generated FSA for the modeled SRP non-malicious node, shown in Appendix A, contains 28 states.

**SRP model validation.** Following the validation criteria formed in Section 6.3, we validate the SRP model to ensure the paths are correctly constructed and that the cryptographic MAC allows the initiator to check the *rrep* to ensure it matches the MAC value the target computed over the accumulated path in the *rreq*.

The first validation step is to ensure the paths are constructed according to the SRP protocol. Figure 33 contains SPIN generated output for the three returned paths using the network validation topology in Figure 29 on page 61, with initiator node 0 and target node 4.

---

**Path 0-4:**

- from _wm_local?rrep,0,4,0.4,255,255,255,0.0,4,255,255,255
- MAC Validated
- Route Discovered

**Path 0-1-2-4:**

- from _wm_local?rrep,0,4,0.1,2,4,255,0.0,1,2,4,255
- MAC Validated
- Route Discovered

**Path 0-1-2-3-4:**

- from _wm_local?rrep,0,4,0.1,2,3,4,0.0,1,2,3,4
- MAC Validated
- Route Discovered

---

**Figure 33. SRP Path Validation**

The message format is identical to the DSR model, with the addition of appending the modeled MAC computation (*mac_path*) to the end of the message. The bold underlined portions indicate the *accum_path* and *mac_path* respectively. These values must match for the SRP initiator node to accept the route.

The second step in the validation process to ensure the MAC over the accumulated path in the *rrep* is properly modeled. In SRP, the MAC ensures that changes to the path during the *rrep*, which occur after the target signs the path constructed in the *req*, are detected by the initiator and the route is discarded. We check the modeled MAC by changing the accumulated path during the *rrep*. If the MAC is modeled properly, the initiator node should detect these direct path changes and reject the route. We use the topology in Figure 34 to validate the MAC, where node 0 is the initiator, node 3 is the target, and node 4 is the malicious node. In Promela models that contain attackers, all non-malicious nodes are initiated first, followed by initiating the attacker nodes. During validation runs, the attacker node deletes a node from the return *rrep* path without attempting corresponding changes to the MAC value. The MAC value should indicate the return path has been corrupted.
Figure 34. Five-node Attacker Topology

Figure 35 contains SPIN generated output where the malicious node 4 changes the accumulated path during the *rrep*. The bold underlined portions, indicating the `accum_path` and `mac_path` respectively, do not match, thus the initiator node 0 does not accept the route. The initiator’s ability to detect the attack gives us confidence that the MAC is modeled correctly. We additionally analyze this attack against all possible paths for the given topology using the SPIN exhaustive evaluation method, which does not find any cases where SRP accepts corrupted routes. While SRP detects and avoids this simple path attack against the *rrep*, SPIN exhaustive analysis allows us to detect that SRP is susceptible to attacks when the attacker corrupts the path during the forward *rreq*, prior to the target’s MAC computation. In Chapter 7, we demonstrate SPIN’s ability analyze the protocol and attacker models to detect successful attacks against the forward *rreq* in SRP.

Path 0-1-2-4-3:
```
from_wm_local?rrep,0,3,0.1.4.3,255,0.0.1.2.4.3
MAC Failed
```

Figure 35. SRP MAC Validation

6.4.3 Modeling Ariadne

Ariadne [24] is an extension to the DSR protocol that attempts to secure the forward *rreq* by computing a one-way per-hop hash value at each intermediate node and attempts to secure the return *rrep* by attaching a target signature computed over the accumulated forward path.

The Ariadne message formats are:
- `<rreq, initiator, target, id, hash_value, accum_path, sig_list>`
- `<rrep, target, initiator, accum_path, sig_list, target_sig>`.

We illustrate the Ariadne protocol using the network topology and message sequence from Figure 36. Node 0 is the initiator, node 3 is the target, *H* is a cryptographically secure one-way hash function, and *SK* is node *i*'s signing key.
The hash value \((h_x)\) is incorporated to guard against an attacker removing a node from the embedded path. The initial hash seed is based on a secret known only to the initiator-target pair. Each intermediate node adds its id to the route path \((\text{accum\_path})\), calculates and inserts a new per-hop hash value \((\text{hash\_value})\), and appends its signature before retransmitting the \textit{rreq}. In addition to the hash value, each intermediate node computes and appends a signature \((\text{sig\_list})\) over the complete packet to ensure the path contains only trusted insiders. Ariadne allows the signature to be a message authentication code (MAC) computed with a pairwise secret key, a MAC computed with a delayed key via the Timed Efficient Loss-tolerant Authentication (TESLA) [116] broadcast key distribution scheme, or computed as a public signature. We use the public signature scheme and refer to node \(x\)'s signature as \(\text{sig}_x\).

Once the target receives the \textit{rreq}, it validates the one-way hash value by iteratively performing the hash computation against all nodes in the accumulated path. If the \textit{rreq} is valid, the target signs the path and returns the path and signature in the \textit{rrep}. During the \textit{rrep}, the intermediate nodes along the unicast path relay the reply to the next upstream node indicated in the embedded path. The initiator checks the signatures and accepts the route if all checks validate.

### Ariadne model development

We base the Ariadne model development on the high level description over the Ariadne route discovery process in Figure 37 below, derived from the Ariadne message format and the Ariadne protocol from Figure 36.

The first step to translating Ariadne into a Promela model is to define the message format. Following the DSR model development, the identifier \((id)\) is dropped from the message since we are analyzing one route discovery round. We replace the \textit{id} with a Boolean value in each node to ensure only one \textit{rreq} is processed during the route discovery attempt. To maintain common message formatting and subsequent channel modeling, the \textit{rreq} and \textit{rrep} messages are modeled following the same format. The intermediate and target nodes signatures are combined, producing the following interim message structure:

\[
<\text{msg\_type}, \text{initiator}, \text{target}, \text{accum\_path}, \text{hash\_chain}, \text{sig\_list}>.
\]

Ariadne does not use the \textit{hash\_chain} in the \textit{rrep} and it is set to zero after the target processes the \textit{rreq}. The two fields of interest are the \textit{hash\_chain} and the \textit{sig\_list}, which respectively model the cryptographic one-way hash value and the list of cryptographic signatures.
**Figure 37. High Level Ariadne Route Discovery Process**

The hash chain represents a cryptographically secure one-way computation. At each stage, an intermediate node rehashes the hash chain after appending its node id to the previous hash value. This process produces the following one-way hash chain, with \( i \) representing the current node and \( i-1 \) representing the previous node along the forward \( rreq \):

\[
h_i = H[n_i, h_{n_{i-1}}] = H[H[n_i, H[n_{i-1}, \ldots H[n_0, h_0]]].
\]

During protocol operation, the recomputed hash chain is transmitted in each message. For modeling purposes we remove the hash operation, listing the hash chain as: \([n_i, n_{i-1}, \ldots, n_1, n_0]\). Modeled nodes may not corrupt or remove an earlier appended value to the received hash chain directly, since we are modeling a one-way hash value based on the chain’s node identifiers in the appended array. We capture the hash computation by restricting a node’s actions against the modeled hash chain to either replay the entire chain or compute a hash value by appending to
any previous hash chain captured from the wireless environment for the given route discovery round.

We represent the hash chain in Promela with an append only accumulated array. To capture the properties of the one-way hash computation, the model does not allow an intermediate node to change the hash_chain. The modeled hash chain is different than the accumulated path (accum_path), as the model does not restrict changes to the accumulated path. Once the target node receives the rreq, it checks the accumulated path against the hash chain to ensure that they match. This process captures the effect of the target computing a one-way hash value using the path contained in the rreq message and comparing it against the hash value delivered in the rreq.

The Ariadne accumulated intermediate node signatures (sig_list) protect the protocol from including malicious outsiders in the routing path. The initiator validates the signatures against the received accumulated path. We model the sig_list in the same fashion as the hash value, appending a node id to the sig_list to indicate a node has signed the given message. The signature list (sig_list) is modeled by an array, whereby the last entry indicates a signature is calculated over the message and all previous signatures. For instance, the signature field in msg_3 from Figure 36 on page 66 is (sig_1, sig_2), where the sig_2 computation contains sig_1. To capture the signature’s cryptographic properties, we do not allow the modeled signatures to be reordered, but they can be dropped in the reverse order to match any attacker dropping nodes from the end of the accumulated path. Assuming the signature process is cryptographically secure to polynomial bounded attackers and limiting the attacker process to dropping nodes from the end of the path, we do not explicitly model the intermediate signatures during the forward rreq in order to reduce the state-space. Any node deleted from the end of the accumulated path is inherently matched by dropping the corresponding appended signature.

We maintain Ariadne’s security mechanisms by focusing the model on the per-hop one-way hash value during the forward rreq and the target signature over the path to secure the embedded path returned in the return rrep. The target models the signature by copying the accumulated path from the forward rreq into the signature in the rrep. We do not allow the modeled cryptographically secure signature to be corrupted during the rrep, allowing the initiator to check the returned accumulated path against the target signature in order to ensure routing attacks performed against the return rrep are detected.

The Ariadne model therefore uses the following message format, where accum_pos and hash_pos track the current position for the non-malicious node to append its id to the accumulated path and modeled hash chain to the current forward rreq message:

<msg_type, initiator, target, accum_path, accum_pos, crypt_val, hash_pos>.

We use a single array called crypt_val to model both the hash chain and target signature. During the forward rreq, crypt_val holds the hash chain. During the return rrep, crypt_val holds the target signature.
The resulting SPIN generated FSA for the modeled Ariadne non-malicious node, shown in Appendix A, contains 32 states.

**Ariadne model validation.** Following the validation criteria formed in Section 6.3, we validate the Ariadne model to ensure the paths are correctly constructed, the per-hop hash value protects the path in the forward *rreq*, and the target signature protects the reverse *rrep*.

The first validation step is to ensure the model constructs paths according to the Ariadne protocol. Figure 38 contains SPIN generated output for the three returned paths using the network validation topology in Figure 29 on page 61, with initiator node 0 and target node 4.

---

**Path 0-4:**

```
from_wm_local|rreq,0,4,0,255,255,255,255,255,1,0,255,255,255,255,1
rreq received at target
Target validated hash
```

```
from_wm_local|rrep,0,4,0,255,255,255,0,255,255,255,0
rrep received at initiator
Initiator validated target signature
Route discovered
```

**Path 0-1-2-4:**

```
from_wm_local|rreq,0,4,0,1,2,255,255,3,0,1,2,255,255,3
rreq received at target
Target validated hash
```

```
from_wm_local|rrep,0,4,0,1,2,4,255,0,0,1,2,4,255,0
rrep received at initiator
Initiator validated target signature
Route discovered
```

**Path 0-1-2-3-4:**

```
from_wm_local|rreq,0,4,0,1,2,3,255,4,0,1,2,3,255,4
rreq received at target
Target validated hash
```

```
from_wm_local|rrep,0,4,0,1,2,3,4,0,0,1,2,3,4,0
rrep received at initiator
Initiator validated target signature
Route discovered
```

---

**Figure 38. Ariadne Path Validation**

The message after the receive operator (?) follows the modeled Ariadne message format. The bold underlined portions indicate the modeled path and hash structures respectively for the *rreq* seen at the target and the path and target signature respectively for the *rrep* seen at the initiator.

The next steps in the validation process are to ensure the modeled forward per-hop hash value detects changes to the *rreq* and that the modeled target signature detects changes to the *rrep*. For both validation steps, we use the network topology in Figure 34 on page 65. For the per-hop hash validation, we construct an attacker that deletes a node from the path during the forward *rreq*. SPIN evaluation executes the model, showing the target node detects and avoids
the attack since the path and modeled hash chain do not match, as indicated by the bold underlined portions respectively shown in the \textit{rreq} validation output in Figure 39.

To validate the target signature, we construct an attacker that changes the path during the \textit{rrep}. SPIN evaluation executes the model, showing the initiator detects and avoids the attack. The initiator detects the attack since the path and modeled target signature do not match, as indicated by the bold underlined portions respectively shown in the \textit{rrep} validation output in Figure 39. For both attackers, we also use SPIN to exhaustively analyze all route possibilities in the given scenario and do not discover any cases where Ariadne accepts corrupted routes, indicating the modeled security features perform as intended. While Ariadne catches these simple attacks, we show in Chapter 7 SPIN's ability to analyze the protocol and a more sophisticated attacker model to detect and demonstrate successful attacks against the Ariadne route discovery process.

---

**Validating the \textit{rreq} at the target:**

\textit{Path 0-1-2-4-3 at target:}

\begin{verbatim}
from_wm_local?rreq,0,3,0,1,4,255,255,3,0,1,2,255,255,3
Hash failed
\end{verbatim}

**Validating the \textit{rrep} at the initiator:**

\textit{Path 0-1-2-3 at target:}

\begin{verbatim}
from_wm_local?rrep,0,3,0,4,3,255,255,0,0,1,2,3,255,0
Signature failed
\end{verbatim}

---

**Figure 39. Ariadne Hash and Signature Validation**

### 6.4.4 Modeling endairA

The endairA protocol [45] attempts to secure DSR by securing only the return \textit{rrep} using cryptographic signatures. The endairA forward \textit{rreq} is identical to DSR, using no cryptographic mechanisms to secure the \textit{rreq}. This approach is different than SRP’s attempt to secure the forward \textit{rreq} process and Ariadne’s attempt to secure both the forward \textit{rreq} and return \textit{rrep} processes.

The endairA message formats follow as:

- \textit{<rreq, initiator, target, id, accum\_path>}
- \textit{<rrep, initiator, target, accum\_path, sig\_list>}

We illustrate the endairA protocol using the network topology and message sequence shown in Figure 40. Node 0 is the initiator, node 3 is the target, and \(SK_i\) is node \(i\)’s signing key. Instead of protecting the forward \textit{rreq} process, the target computes a signature over the accumulated path received in the \textit{rreq} and adds the signature to the \textit{rrep}. During the \textit{rrep}, the intermediate nodes sign the message and forward to the next hop. Once the \textit{rrep} reaches the initiator, the initiator checks the target signature and verifies that each node in the return path has signed the message in reverse order. While the target may sign corrupted paths received by the \textit{rreq}, the
protocol authors contend that false paths should not be successfully returned to the initiator with the correct appended signatures.

\[
\text{msg}_0 = (\text{rreq}, 0, 3, \text{id}, ( )) \\
\text{msg}_1 = (\text{rreq}, 0, 3, \text{id}, (1)) \\
\text{msg}_2 = (\text{rreq}, 0, 3, (1, 2)) \\
\text{msg}_3 = (\text{rep}, 0, 3, (1, 2), \{\text{sig}_3\}) \\
\text{sig}_3 = \text{SK}_3\{\text{rrep}, 0, 3, \text{id}, (1, 2), ()\} \\
\text{msg}_4 = (\text{rep}, 0, 3, (1, 2), \{\text{sig}_3, \text{sig}_2\}) \\
\text{sig}_2 = \text{SK}_2\{\text{rrep}, 0, 3, (1, 2), (\text{sig}_3)\} \\
\text{msg}_5 = (\text{rep}, 0, 3, (1, 2), \{\text{sig}_3, \text{sig}_2, \text{sig}_1\}) \\
\text{sig}_1 = \text{SK}_1\{\text{rrep}, 0, 3, (1, 2), (\text{sig}_3, \text{sig}_2)\}
\]

Figure 40. endairA Protocol Messages

endairA model development. We base the endairA model development on the high level description over the endairA route discovery process in Figure 41 below, derived from the message format and the endairA protocol messages from Figure 40.

\textit{rreq} process:
- Initiator node
  - Initiate a \textit{rreq} to target
- Intermediate nodes
  - If previously seen \textit{rreq} → take no action
  - Else
    - If not target → append \text{id} to path and retransmit \textit{rreq}
    - If target → take actions below in \textit{rrep}

\textit{rrep} process:
- Target node
  - Calculate and attach signature over the path in the received \textit{rreq}
  - Unicast the \textit{rrep}
- Intermediate nodes (along the unicast path)
  - If not initiator
    - Calculate and attach signature over the received \textit{rrep}
    - Transmit updated \textit{rrep} to next upstream host
  - If initiator
    - Validate the accumulated path against the target signature
    - Validate individual signatures to ensure that every node in target signature has supplied a signature in the reverse path order

Figure 41. High Level endairA Route Discovery Process

The approach to modeling endairA is similar to the previous source routing protocol models. Again, we drop the \text{id} from the message and replace it with a Boolean value to ensure only one \textit{rreq} is processed during an individual route discovery attempt. To maintain common message
formatting and subsequent channel modeling, the \textit{req} and \textit{rep} messages are represented following the same format. The modeled endairA message format follows as:

\texttt{<msg\_type, initiator, target, accum\_path, accum\_pos, target\_sig, sig\_list, sig\_pos>}

The variable \textit{target\_sig} holds the signature over the accumulated path that the target constructs and embeds into the \textit{rep}. The target signature is intended to protect the path from being corrupted during the \textit{rep}. The value is modeled by copying the \textit{accum\_path} array into the \textit{target\_sig} array. In order to capture the target signature's cryptographic properties, we do not allow the modeled intermediate nodes to change the \textit{target\_sig} value. The signature list (\textit{sig\_list}) keeps track of each node that has signed the return \textit{rep} during its message traversal. We express this activity in an array, where each node adds its id to the \textit{sig\_list} during the \textit{rep}. Once the initiator receives the \textit{rep}, it checks the target signature to ensure the accumulated path matches the signature. The initiator also verifies that all nodes in the accumulated path have signed the message by checking the signature list in the reverse order, since the return signatures are added in the opposite order than the forward route construction.

The resulting SPIN generated FSA for the modeled endairA non-malicious node, shown in Appendix A, contains 34 states. 

\textbf{endairA model validation.} Following the validation criteria formed in Section 6.3, we validate the endairA model to ensure that the paths are correctly constructed, the target signature protects the reverse \textit{rep}, and the intermediate node signatures are appended in the proper order during the \textit{rep} and compared against the signed accumulated path.

The first validation step is to ensure the model constructs paths according to the endairA protocol. Figure 42 contains SPIN generated output for the three returned paths using the network validation topology in Figure 29 on page 61, with initiator node 0 and target node 4.

\begin{verbatim}
Path 0-4:
 from_wm_local?rrep,0,4,0,4,255,255,255,0,4,255,255,255,4,255,255,255,255,255,255,1
 Sig OK
 No Missing Node Sigs
 Route Discovered

Path 0-1-2-4:
 from_wm_local?rrep,0,4,0,1,2,4,255,0,1,2,4,255,0,4,2,1,255,255,3
 Sig OK
 No Missing Node Sigs
 Route Discovered

Path 0-1-2-3-4:
 from_wm_local?rrep,0,4,0,1,2,3,4,0,1,2,3,4,0,4,3,2,1,255,4
 Sig OK
 No Missing Node Sigs
 Route Discovered
\end{verbatim}

\textbf{Figure 42. endairA Path Validation}
The message format after the receive operator (?) follows the modeled endairA message structure. The bold underlined portions indicate the accumulated path, the target signature over the path, and the return signature list respectively.

The next steps in the validation process are to ensure the modeled target signature detects changes to the \texttt{rrep} and that the modeled signature list ensures all nodes in the identified path have supplied their signatures in the correct order. For both validation steps, we use the network topology in Figure 34 on page 65. To validate the target signature process, we construct an attacker that corrupts the path during the return \texttt{rrep}. SPIN evaluation executes the model, showing the initiator successfully detects and avoids the attack. The initiator does not accept the route, since the modeled path and target signature do not match, as indicated by the bold underlined portions respectively as shown in Figure 43.

To validate the intermediate node signature check, we construct an attacker that removes an intermediate node signature during the \texttt{rrep}. SPIN evaluation executes the model, showing the initiator successfully detects and avoids the attack. The initiator does not accept the route since the accumulated path and signature list, which is in reverse order, do not match as shown in Figure 43. For both attackers we also use SPIN to exhaustively analyze all route possibilities in the given scenario and do not discover any cases where endairA accepts corrupted routes, indicating the modeled protocol performs as intended. While endairA detects and avoids these simple attacks, we show in Chapter 7 SPIN's ability to analyze the protocol and a more sophisticated attacker model to detect and demonstrate attacks against the endairA route discovery process.

Validating the target signature:

\begin{verbatim}
Path 0-1-2-3:
    from_wm_local?rrep,0,3,0,1,2,3,255,0,1,2,3,255,0,3,4,255,255,255,2
Signature failed
\end{verbatim}

Validating the intermediate node signature order:

\begin{verbatim}
Path 0-1-2-3:
    from_wm_local?rrep,0,3,0,1,2,3,255,0,1,2,3,255,0,3,4,255,255,255,2
Signature OK
    Missing Node Signature
\end{verbatim}

\textbf{Figure 43. endairA Signature Validations}

\section{6.5 Modeling the Attacker}

There are attacks against MANET routing protocols that have no known detection or prevention mechanism. Attacks such as the invisible node attack (INA) [44,94] are a continued threat, since no current mechanisms can positively identify the node that transmitted a given message [97]. Additionally, malicious insiders can refuse to participate in routing protocols at will, even though they are trusted to follow the protocol rules.
This research focuses on an attacker that actively corrupts the embedded route during the MANET route discovery phase, resulting in routes that do not match the current network topology. To produce a finite model space, we focus on attacker actions to drop a node from the accumulated path during the route discovery process. Evaluating a node’s ability to actively corrupt the routing messages to delete nodes from the routing path is a common practice \cite{21,45,54,67,68}. For the remainder of this thesis, we term this attacker activity as the *node drop attack* (NDA).

We isolate the attacker actions into a separate Promela process type. Using a modularized approach simplifies other adaptations to the attacker capabilities and goals against the various protocols. Each attacker requires a different attacker process, so when modeling a different attacker, the attacker process is simply replaced. The attacker cannot break cryptographic mechanisms, but is not forced to follow the routing protocol operations.

Developing the attacker model relies on static analysis techniques to produce a finite attacker model that an automated model checker can execute. If we allowed an attacker to arbitrarily change messages, the exhaustive state-space consisting of all possible messages would quickly exceed computational capabilities. The routing messages and structure are dependent on the message interactions between the intermediate nodes, based on the network topology. For instance, the number of nodes included in a route depend on the network topology and the source and destination locations in the topology. Exhaustively attempting all message formats without knowing what information may enable the attack requires all possible message orderings for each possible path between each possible source-destination pair. Modeling the MANET routing process through multiple intermediate nodes requires a more complicated analysis structure compared to the simple three node authentication protocol analysis structure illustrated in Figure 16 on 38.

We leverage the fact that a simplified three node structure, consisting of a source, destination, and attacker, can model end-to-end authentication protocol security requirements \cite{66}. Rather than using model checking to generate all possible message structures for a simple three node structure, we evaluate all possible path sequences through the wireless network topology for a given sized network. To develop the attacker model we follow the approach in \cite{108} to limit the attacker’s actions based on the information that may be revealed by the protocol messages, which could possibly enable an attack. This process requires static analysis over the possible messages the attacker can capture and extracts only the information required by the attacker to perform a successful attack. The information obtained by the static analysis allows the attacker to model a finite set of attempted attack sequences. The *pre-analysis* is vital in translating the attacker into a model checker, as the attacker cannot attempt all possible message combinations within the constraints of a finite state model checker.
In the remainder of this section, we focus on the NDA model development for the SRP, Ariadne, and endairA protocols.

### 6.5.1 INA Attack Development

While we focus attacker development on the NDA, we first model the invisible node attack (INA) to ensure the SPIN model checker can incorporate attacker nodes with the modeled non-malicious nodes. Modeling the INA allows us to ensure the attackers receive all possible messages, which may determine if attacks are possible. We must ensure the attacker nodes can receive messages during the unicast `rrep` even if they are not listed in the source routing path. Recall that to reduce the modeling state-space, we modeled the wireless medium server to deliver the unicast `rrep` only to the next upstream node and to any attacker nodes within the transmitting node’s communication footprint, as opposed to sending all `rrep` packets to all local neighbors. Since the INA is not listed in the source routing path, the wireless medium server must deliver `rrep` messages to the attacker node even though the node is not listed in the unicast message. Demonstrating that the INA is successful during SPIN evaluations gives us confidence that attacker nodes receive information transmitted within their reception range.

The INA [44,94] occurs when an attacker refuses to add its own id to the routing path during route discovery. The attacker model relays the forward `rreq` message without adding its id to the accumulated path and relays any subsequent `rrep`. As an example, a SPIN generated INA attack against the network topology in Figure 34 on page 65 results in the path `0-3` being returned instead of the actual path `0-4-3`. Even though the attacker is not part of the unicast `rrep`, the modeled wireless medium ensures the attacker receives any wireless message transmitted within its reception range. The attacker does not expect to be listed in accumulated path and simply relays the `rrep` to next upstream node in the accumulated path.

The resulting SPIN generated FSA for the modeled INA, shown in Appendix A, contains seven states.

### 6.5.2 SRP Attack Development

The INA may be intuitively described as a relay attack. Conversely, the NDA occurs when an attacker corrupts a route by proactively dropping a node from the accumulated routing path embedded into the route discovery messages. We use static analysis to determine which actions need to be included in the NDA to attack the SRP model. Recall the actions SRP takes to protect the accumulated path against routing attacks is for the target to calculate a MAC over the path extracted from the `rreq` and to include the MAC in the `rrep`. Attacks corrupting the `rrep` are detected by SRP, as discussed in the SRP model validation, since the MAC value does not match the corrupted path. Therefore, the goal of the SRP attacker is to corrupt the forward `rreq` process to trick the target into computing and returning a MAC over a corrupt route. The target
computes a MAC over any path that is delivered within the *rreq*, to include paths that have been corrupted and do not contain valid routes.

The modeled SRP NDA allows complete automation, without requiring any *a priori* knowledge of the network configuration and does not rely on any manual intervention during SPIN analysis. Figure 44 provides a high level description for the SRP NDA.

<table>
<thead>
<tr>
<th><strong>rreq process:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• If previous node initiator → take no action</td>
</tr>
<tr>
<td>• Else</td>
</tr>
<tr>
<td>o Remove previous node id from accumulated path</td>
</tr>
<tr>
<td>o Add own id to path</td>
</tr>
<tr>
<td>o Retransmit <em>rreq</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>rrep process:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Send unicast message to all upstream nodes</td>
</tr>
</tbody>
</table>

**Figure 44. High Level SRP NDA Process**

During the *rreq* process the attacker removes the previous node from the accumulated path and replaces its own id in the previous node's place, as long as the previous node is not the initiator. Once the corrupted route reaches the target, it is included in the MAC calculation. During the *rrep* process the attacker relays the unicast *rrep* to all upstream nodes from the embedded path, attempting to deliver the corrupted packet to the initiator or a node in the upstream path that can eventually reach the initiator. The attacker cannot use the original dropped node to relay the *rrep* on the way back to the initiator, as the deleted node is no longer part of the embedded unicast path and will not forward *rrep* messages in which it is not included.

The only requirement for the modeled NDA to be successful against SRP is that the attacker must be able to successfully transmit the corrupted path during the *rrep* to any upstream node from the accumulated path. The ability to reach the initiator without using the dropped node depends on the current network topology. Since the intermediate nodes do not utilize any cryptographic mechanisms, the attacker may be an insider or an outsider.

The resulting SPIN generated FSA for the SRP NDA, shown in Appendix A, contains 12 states. Chapter 7 demonstrates the automated attack analysis results.

### 6.5.3 Ariadne Attack Development

We use the messages in Figure 36 on page 66 for static analysis to develop the Ariadne attacker. Since the target node signs the accumulated path received in the *rreq*, the path cannot be corrupted during the *rrep*, as shown in the Ariadne model validation. Thus, the target signature limits the attacker to the forward *rreq*. Examining the *rreq* message structure indicates that the
unprotected accumulated path can be changed at will, as long as the corresponding hash chain and signatures match the path changes. The forward intermediate signatures do not allow the attacker to reorder the path or an outsider to add to the path. Since we are not explicitly modeling the forward signatures, we limit the malicious insider to only appending or dropping a node from the end of the accumulated path. We also limit the malicious outsider to only dropping a node from the end of the accumulated path, since the outsider cannot generate a valid signature without a trusted key.

The final message element to consider is the one-way hash value. Due to its cryptographic properties, the hash value cannot be directly corrupted. Any attacker that attempts to drop a node from the accumulated path must compute a hash value that corresponds to the new path. The only way for an attacker to strip a node from the end of the accumulated path and compute a correct corresponding hash value is to generate the hash value based on capturing a previous hash value from an upstream neighbor during the current route discovery round. Hash values from previous route discovery rounds do not match due to the message id, or sequence number. For instance, if an attacker wishes to remove node 2 from the protocol example in Figure 36 on page 66, it needs to know the value $h_1$. If $h_1$ was not directly captured from $msg_2$ it could be calculated from $h_0$, which is extracted from $msg_1$, since $h_1 = H[1,h_0]$.

**Ariadne NDA.** Since Ariadne requires intermediate nodes to add a signature to the forward $rreq$, NDA attempts where a malicious node drops the upstream packet in place of itself is limited to malicious insiders.

Figure 45 provides a high level description for the NDA against Ariadne. The Ariadne NDA can be performed by a malicious insider.

---

**Figure 45. High Level Ariadne NDA Process**

If attacker receives a $rreq$ directly from the initiator, it stores the hash value and takes no further action. Upon receiving a $rreq$ from an intermediate node, the attacker first stores the associated node's hash value. We model the attacker's storage using the constructs shown in
Figure 46. Since the hash value is modeled as a copy of the current accumulated path at each step, the attacker must be able to store up to $N$ hash chains with a maximum path size of $N$, for a network containing $N$ nodes.

```
Global type definition
typedef Attacker_Store{
    bool h_stored = 0; /* marks if hash is stored */
    byte h_chain[N] = 255; /* holds captured hash chain */
}

Local attacker variable definition
/* initiate a space for up to N hash chains*/
/* hash chain stored in array position corresponding to node that generated the hash*/
Attacker_Store a_store[N];
```

**Figure 46. Modeling Attacker Storage Space**

Once the message is received and the hash chain is stored, the attacker checks if it knows a hash value from a previous upstream host listed in the embedded path. If the attacker has this knowledge, it removes the last node id and adds its own id to the accumulated path. The attacker also replaces the hash value with its local hash calculation. To compute a valid hash, the modeled attacker appends nodes to a previous hash-chain until it matches the current accumulated path. During the rrep, the attacker attempts to relay the rrep to any upstream neighbor identified in the embedded path.

**Ariadne INDA.** We also consider the combination of the INA and NDA attacker, where the attacker actively corrupts the path and makes the corresponding hash changes without adding its id to the routing path. We refer to this attack as an invisible NDA (INDA). The INDA allows a malicious outsider, without cryptographic keys, the ability to attack Ariadne. To the best of our knowledge, the INDA is the first discovered active path corruption attacker against Ariadne by a malicious outsider. In chapter 7 we demonstrate our discovery of this attack using SPIN analysis over the modeled Ariadne protocol and INDA attacker.

The INDA, which can be performed by a malicious outsider, follows the same process as shown for the NDA in Figure 45, except that it does not add its id to the path after dropping the last node from the accumulated path. These attacks are successful as long the attacker lies in a position to capture a previous enabling hash value and the rrep can be delivered to one of the attacker’s upstream neighbors.

The resulting SPIN generated FSA for the Ariadne NDA and INDA, shown in Appendix A, both contain 25 states. Chapter 7 demonstrates SPIN’s exhaustive automated attack analysis for the Ariadne NDA and INDA.
6.5.4 endairA Attack Development

We use the messages in Figure 40 on page 71 for static analysis to develop the endairA attacker. As in Ariadne, the target signature over the accumulated path ensures the path cannot be corrupted during the rrep, since the signatures added by the intermediate nodes during the rrep must match the reverse order as the path signed by the target in the rrep. We target attacks against endairA using malicious insiders. If a malicious node drops a node during the forward rreq, it requires a direct link to the dropped node’s upstream neighbor to ensure the appropriate signatures can be produced in the correct order during the rrep. However, this link constitutes a valid path and does not result in a path inconsistent with the network topology. The only avenue that allows the malicious insider the ability to drop nodes without this direct link is the ability to generate signatures for more than one node. If we assume the cryptographic process is secure in polynomial time, the attacker must have multiple keys to produce multiple signatures. In a colluding environment, we assume that all colluding attackers have previously shared their keying material.

Figure 47 provides a high level description for the NDA against endairA. The endairA NDA can be performed by two colluding malicious insiders. The colluding nodes are aware of each other and share copies of their keys. If the forward rreq passes through two attackers, the second attacker removes the nodes between the two attackers from the path before sending the rreq to the next hop. Once the target signs the accumulated path it responds with the rrep. Note that the attacker tricked the target into signing a corrupted path. During the rrep, the second attacker node signs for both attackers in the correct order to ensure the signatures match the nodes in the accumulated path. The requirement to enable this attack is that the second attacker must be able to relay information to any upstream node prior the first attacker.

---

**rreq process:**
- If any upstream node in accumulated path is an attacker
  - Drop all nodes in path after first attacker
  - Add own id to path
  - Transmit changed rreq
- Else (you are the first attacker)
  - Follow protocol rules and append own id to accumulated path
  - Transmit changed rreq

**rrep process (processed by the second attacker):**
- Apply signatures for self and colluding attacker
- Transmit the changed rrep to any nodes in path upstream to the first attacker

---

**Figure 47. High Level endairA NDA Process**
The resulting SPIN generated FSA for the endairA NDA, shown in Appendix A, contains 17 states. Each colluding attacker asynchronously follows an independent instance of the FSA. Chapter 7 demonstrates the automated attack analysis results.

6.6 Automated Topology Generation and Analysis

The protocol and attacker models allow us to use SPIN to determine if a message sequence exists that allows an attack for a given network topology. One of the biggest impediments to other analysis methods is the intuition to choose a network configuration in which the attack exists. For instance, the attacks indicated by the simulatability model [21,45,54] and subsequently discovered through visual inspection rely on specified network topologies. Nanz's [67,68] preliminary work towards automating security analysis for on-demand distance vector protocols and our own work in [117] using SPIN as a method to automate the security analysis process for on-demand source routing protocol automates analysis over a static, pre-determined network topology.

A complete automated analysis capability should not rely on manually choosing an enabling topology. To complement the Promela protocol and attacker models, we develop exhaustive topology generation, evaluation, and reporting routines. This combination provides an automated security analysis process, as illustrated in Figure 22 on page 50. The overall analysis process combines SPIN’s exhaustive analysis for a static network and ensures that all network topologies for a given sized network are evaluated. The evaluation process consists of specifying the network size (N) and evaluating the protocol and attacker model against each possible topology. The process relies on duplicating the SPIN Promela file for each topology by adapting the network connectivity array to reflect each configuration.

6.6.1 Calculating the Topology Space

We use the connectivity array in Figure 48 to aid in describing the process to calculate the total topology space. When modeling bidirectional networks, the shaded array areas indicate the symmetry between the corresponding network links.
We first determine the number of topologies for a directional network. In a directional network, the links are one-way, or asymmetric. For asymmetric networks, we do not consider the symmetric areas in Figure 48 to determine the total possible topologies. In the asymmetric case, each row contains $2^{(N-1)}$ binary combinations. Since we have N rows, the number of topologies is determined by Equation 2:

$$2^{(N-1)} \times 2^{(N-1)} \times \ldots \times 2^{(N-1)} = 2^{N(N-1)}.$$ (2)

When modeling bi-directional networks, we rely on the symmetry between the links to determine the number of topologies. The number of topologies is based on the possibilities for the shaded areas to one side of the zero diagonal in Figure 48. Since the number of elements that determine the symmetric topology are half that of the asymmetric case, we calculate the number of symmetric topologies using Equation 3:

$$2^{N(N-1)/2}.$$ (3)

Following Equations 2 and 3, Table 6 indicates the total unique network topologies that must be generated and evaluated to ensure all possible message event sequences for a given network size $N$. These topologies range from fully disconnected, with no links, to fully connected, where each node connects to all other nodes.

<table>
<thead>
<tr>
<th>$N$</th>
<th>Symmetric</th>
<th>Asymmetric</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>64</td>
<td>4,096</td>
</tr>
<tr>
<td>5</td>
<td>1,024</td>
<td>$\sim 1.049 \times 10^5$</td>
</tr>
<tr>
<td>6</td>
<td>32,768</td>
<td>$\sim 1.074 \times 10^5$</td>
</tr>
</tbody>
</table>

During analysis, the number of network topologies generated and searched must be balanced against the possibility that existent attacks are identified. If no attacks are discovered for a small network size, the protocol must be analyzed using a larger network or be mathematically proven that the security property extends to larger network sizes. In the protocol analysis demonstrated in Chapter 7, we successfully find attacks against all the modeled protocols using the network size $N = 5$, assuming the network is bidirectional and has symmetric links. Therefore, we scope the remainder of the discussion for a 5-node symmetric network, totaling 1,024 unique topologies, according to Equation 3.

Now that we know how many topologies are possible, we focus on generating the topologies, evaluating the topologies, and reporting which topologies produced failures during SPIN protocol analysis.
6.6.2 Topology Generation Engine

The Topology Generation Engine generates a new Promela file for each possible network topology for the given network size. The 5-node symmetric topology array is illustrated in Figure 49.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>val-9</td>
<td>val-8</td>
<td>val-7</td>
<td>val-6</td>
</tr>
<tr>
<td>1</td>
<td>val-9</td>
<td>0</td>
<td>val-5</td>
<td>val-4</td>
<td>val-3</td>
</tr>
<tr>
<td>2</td>
<td>val-8</td>
<td>val-5</td>
<td>0</td>
<td>val-2</td>
<td>val-1</td>
</tr>
<tr>
<td>3</td>
<td>val-7</td>
<td>val-4</td>
<td>val-2</td>
<td>0</td>
<td>val-0</td>
</tr>
<tr>
<td>4</td>
<td>val-6</td>
<td>val-3</td>
<td>val-1</td>
<td>val-0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 49. 5-Node Symmetric Topology Array Values

At implementation, we represent the upper (or lower) half of the array with a binary string value as illustrated in Figure 50.

<table>
<thead>
<tr>
<th>val-9</th>
<th>val-8</th>
<th>val-7</th>
<th>val-6</th>
<th>val-5</th>
<th>val-4</th>
<th>val-3</th>
<th>val-2</th>
<th>val-1</th>
<th>val-0</th>
</tr>
</thead>
</table>

Figure 50. Binary String Representation

We iterate the binary string through all values to generate each possible topology, starting with the base topology that stores the initial network as fully disconnected. Therefore, all links in the base topology are set to 0. When \( N = 5 \), the binary string is enumerated between 0 and 1023 to represent all possible topologies. For each network topology we read the fully disconnected base SPIN file stored in filename.pml and adjust the network connectivity array as appropriate, saving the output as filename_case-#.pml. For example, the 5-node attacker topology in Figure 34 on page 65 corresponds to the binary string representation: 1001100111 (decimal case 615 resulting in filename_case-615.pml). In this case, val-0 is set to one, representing the least significant bit (rightmost value) from the binary string representation. Once the link value in the topology array is stored for val-0, the binary string representation is shifted right. Now the least significant bit represents val-1 and the link values are stored to the appropriate array locations, followed by right shifting the binary string representation. This process is iteratively followed until the binary string representation for the given case is exhausted. The topology generation engine is summarized in Figure 51.
♦ Set topology in the base SPIN file to fully disconnected
♦ For each individual topology (cases 1 – total cases)
  • Read in the base SPIN file
  • Convert the case # to a binary string representation
    o For each bit in the binary string representation
      ▪ Store the least significant bit to the respective connectivity array locations
      ▪ Shift the binary string
    o Store the current topology in a unique SPIN file

---

**Figure 51. Topology Generation Process**

In the asymmetric case, we similarly generate topologies using a binary string representation for values \( val-0 \) to \( val-19 \) and determine the total number of unique topologies according to Equation 2 on page 81.

### 6.6.3 Evaluation Engine

After topology generation, the local working directory now contains an individual SPIN Promela file containing the given protocol and attacker model for each generated topology. Each unique file must be individually processed with SPIN. To automate this step, we develop an Evaluation Engine, which compiles and executes each individual Promela file for SPIN exhaustive analysis. Successful attack details that result from each file evaluation are generated by SPIN and stored in unique SPIN trail files. Each trail file captures an event sequence leading to the attack. Figure 52 provides an overview for the Evaluation Engine process.

♦ For each file in the current directory
  • If the file is a Promela file (.pml) and not yet processed
    o Generate a SPIN exhaustive analysis file (spin –a filename.pml)
    o Convert the analysis file into an executable (gcc –o pan –CompileTimeOptions pan.c)
    o Run the executable (/pan)

---

**Figure 52. Evaluation Engine Process**

For each Promela file that was generated by the topology engine, the Evaluation Engine compiles the model into an exhaustive analysis routine using SPIN (\( \text{spin} \ –a \ \text{filename.pml} \)) to generate the resulting analysis code. The second step compiles the exhaustive analysis code into an executable SPIN verifier. The last step runs the produced executable to initiate the SPIN analysis process.
6.6.4 Reporting Engine

The final step in the automated security analysis process is the Reporting Engine. The Reporting Engine reads each error file (*.trail) and associated Promela model file (.pml) to report the discovered attacks. Each detected attack is described in terms of the initiator, target, and attacker nodes, along with the corrupted path and the current network topology. The example failure given in Figure 53 displays the automated output failure lines corresponding to an INA attack against DSR identified for the 5-node attacker topology in Figure 34 from page 65. Once an attack is reported, the associated trail file for the respective case can be loaded to view SPIN’s automatically generated event sequence leading to the attack, as we demonstrate in Chapter 7.

<table>
<thead>
<tr>
<th>Processing case: dsr_in_5node_615.pml</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiator Node: 0</td>
</tr>
<tr>
<td>Target Node: 3</td>
</tr>
<tr>
<td>Attacker Node: 4</td>
</tr>
<tr>
<td>Corrupted Path: 0 - 3</td>
</tr>
<tr>
<td>The topology is:</td>
</tr>
<tr>
<td>net_con[0].to[0] = 0 net_con[0].to[1] = 1 net_con[0].to[2] = 0</td>
</tr>
</tbody>
</table>

Figure 53. Automated INA Attack Discovery

6.7 Chapter Summary

In this chapter we describe an automated model checking technique to evaluate route corruption attacks against the route discovery phase for on-demand source routing protocols. The automated analysis process uses the SPIN model checker to examine all message event sequences for a given topology. SPIN background is provided in Section 6.1 and our SPIN development models are contained in Sections 6.2 - 6.5. In Section 6.6 we give an exhaustive topology generation capability to ensure all possible network topologies for a given size network are evaluated. Each topology is subsequently analyzed with SPIN, identifying any configuration and corresponding event sequence leading to an attack.

The automated model checking capabilities combined with the automated exhaustive topology generation, execution, and reporting routines provides a fully automated approach to evaluate security properties in MANET routing protocols. In Chapter 5, we met the thesis objective to develop an adaptive threat model. In this chapter we met the final thesis objective through the automated security analysis process.
CHAPTER 7
EVALUATION CRITERIA AND MODEL DEMONSTRATION

In this chapter we utilize the SPIN Promela models and automated security analysis process developed in Chapter 6 within the adaptive threat framework from Chapter 5 to demonstrate a practical application to evaluate security properties in ad hoc routing protocols.

The remainder of this chapter is outlined as follows: Section 7.1 provides the analysis criteria, Section 7.2 contains the model demonstration, and Section 7.3 discusses modeling computational requirements.

7.1 Analysis Criteria

We define the desired security property ($\varphi$) as a safety criteria that specifies a result that should never exist. The analysis criterion is therefore defined as:

$$\varphi = \text{returned routes never contain a path that does not exist in the network topology.}$$

SPIN exhaustively searches for failures according to the desired security property $\varphi$. That is, to show the routing protocol guarantees the desired security property, we must show that $M$ models $\varphi$, denoted by $M \models \varphi$. This property is formally expressed by Equation 1 on page 52, which indicates that in order for a system to model the given property, all $\sigma$ runs, or possible event sequences, in the system model $M$ must satisfy $\varphi$. To isolate failures, SPIN searches for a contradiction to Equation 1. Therefore, the model checker searches for a failure according to Equation 4:

$$\neg(M \models \varphi) \leftrightarrow \exists \sigma, (\sigma \in M \land \neg(\sigma \models \varphi)).$$

Equation 4 indicates a protocol is not secure if there is at least one path where the system property fails. During exhaustive analysis over a given network topology, SPIN terminates upon the first failure of the property $\varphi$ and produces a counterexample, providing an automatically generated attack sequence.

To evaluate the property $\varphi$ in SPIN, we add a path analysis check. The returned path is checked to ensure each that link exists in the model’s connectivity array. If any link does not exist, an assertion violation is raised and SPIN halts execution, creating a trail file that lists the event sequence leading to the failure. For example, Figure 54 shows how an invisible node attack sequence produces a path that is not consistent with the topology check, since no link exists between node 0 and node 2.
We now demonstrate how the SPIN models and automated analysis process can determine if a given security property $\varphi$ holds as defined for three source routing protocols: SRP, Ariadne, and endairA. Attack development and analysis against each of these protocols highlights a different capability in the automated analysis process.

### 7.2.1 Experimental Details

Table 7 lists the software (SW) and hardware (HW) components used for the model demonstration. Version information for gcc is included since SPIN relies on gcc to compile the exhaustive analysis executable `pan.exe`. Version information for Perl is included since it is used in the Topology Generation, Reporting, and Analysis engines. All SPIN Promela models and Perl network topology generation and analysis scripts are available at the associated web site.

<table>
<thead>
<tr>
<th>Component</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>HW</td>
<td>Intel Pentium 4 3.0-GHZ dual processor with 512MB cache</td>
</tr>
<tr>
<td>Memory</td>
<td>HW</td>
<td>1 GB physical memory</td>
</tr>
<tr>
<td>Operating System</td>
<td>SW</td>
<td>Linux - Fedora Core 6 (2.6.19-1.2895.fc6)</td>
</tr>
<tr>
<td>SPIN</td>
<td>SW</td>
<td>Version 4.2.8</td>
</tr>
<tr>
<td>Perl</td>
<td>SW</td>
<td>Version 5.8.8</td>
</tr>
<tr>
<td>gcc</td>
<td>SW</td>
<td>Version 4.1.1</td>
</tr>
</tbody>
</table>

We set the number of network nodes to $N = 5$ and assume that all nodes assume bidirectional, or symmetrical, links. Assuming bidirectional communication links is common to the MANET development process and associates the current illustration to attacker categories I, II, and VII, as defined in the attacker classification in Table 5 on page 42. However, the modeling approach supports all attacker communication capabilities by changing the topology generation process to exhaustively generate all possible asymmetric topologies according to Equation 2 on page 81.

---

12 Research models and automation routines available at: [http://www.sait.fsu.edu/research/andel](http://www.sait.fsu.edu/research/andel)
7.2.2 Attacking SRP

Static analysis guided SRP attacker development in Section 6.5.2, showing protocol messages may reveal information that allows an attacker to remove a node from the forward rreq during the route discovery process. Since SRP intermediate nodes do not utilize cryptographic mechanisms, the attacker model represents a single malicious outsider (category I).

Attack cases are obtained by running the automated analysis procedure against SRP and the category I attacker. Figure 55 illustrates a single attacker case extracted from the Reporting Engine output file.

<table>
<thead>
<tr>
<th>Processing case: srp_nda-rreq_5node_case-627.pml</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiator Node: 0</td>
</tr>
<tr>
<td>Target Node: 3</td>
</tr>
<tr>
<td>Attacker Node: 4</td>
</tr>
<tr>
<td>Corrupted Path: 0 - 1 - 4 - 3</td>
</tr>
</tbody>
</table>

The topology is:

```
net_con[0].to[0] = 0 net_con[0].to[1] = 1 net_con[0].to[2] = 0 net_con[0].to[3] = 0 net_con[0].to[4] = 1
```

**Figure 55. Automated SRP Attack Case**

Viewing the automatically generated SPIN trail file for case 627 provides a message sequence leading to the SRP NDA. SPIN command line tools textually provide the messages leading to the attack. SPIN also can load the trail file graphically to view the attack sequence. For example, Figure 56 shows the ending of the SPIN attack sequence. The complete SRP NDA attack sequence generated by the trail file is given in Appendix B.

**Figure 56. SRP Automated Attack Sequence**
Following the generated attack sequence above and the SRP model message structure, defined in Section 6.4.2 on page 62, we re-illustrate the attack sequence and associated network configuration in Figure 57.

\[
\begin{align*}
\text{msg}_1 &= (\text{rreq}, 0, 3, Q_{\text{id}}, Q_{\text{sn}}, ( )) \\
\text{msg}_2 &= (\text{rreq}, 0, 3, Q_{\text{id}}, Q_{\text{sn}}, (1)) \\
\text{msg}_3 &= (\text{rreq}, 0, 3, Q_{\text{id}}, Q_{\text{sn}}, (1, 2)) \\
\text{msg}_4 &= (\text{rreq}, 0, 3, Q_{\text{id}}, Q_{\text{sn}}, (1, 4)) \\
\text{msg}_5 &= (\text{rrep}, 0, 3, (1, 4), (MAC_{\text{it}})) \\
\text{MAC}_{\text{it}} &= SA_{\text{it}}\{\text{rrep}, 0, 3, Q_{\text{id}}, Q_{\text{sn}}, (1, 4)\}
\end{align*}
\]

Node 4 drops node 2

\[
\begin{align*}
\text{msg}_6 &= (\text{rrep}, 0, 3, (1, 4), (MAC_{\text{it}})) \\
\text{MAC}_{\text{it}} &= SA_{\text{it}}\{\text{rrep}, 0, 3, Q_{\text{id}}, Q_{\text{sn}}, (1, 4)\}
\end{align*}
\]

Figure 57. SRP Attack Sequence

The attack occurs when Node 4 receives \(\text{msg}_3\) from node 2 containing the current accumulated path 0-1-2. Node 4 removes node 2's id from the accumulated path, adds its own id, and transmits a \(\text{rreq}\) to the target node containing the path 0-1-4 \((\text{msg}_4)\). This activity results in 0-1-4-3 as the reported path between the initiator and target node. Node 3 may have already serviced a \(\text{rreq}\) for the path 0-1-3, since node 3 may have received \(\text{msg}_2\). However, \(\text{rreq}\) messages are not guaranteed deliverability in a wireless environment, so we must consider all possible attack path sequences.

During the route reply process, node 4 attempts to relay the \(\text{rrep}\) in \(\text{msg}_5\) to all upstream nodes contained in the accumulated path. That is, node 4 attempts to send a message to node 0 and node 1. The message for node 0 can be delivered, so the corrupted path 0-1-4-3 reaches node 0. Node 0, who is also the initiator, accepts the corrupted path as valid, since the MAC in the corrupted path matches the MAC computed by the target node.

Since the category I attacker successfully breaks the protocol as a single malicious outsider, all the attacker categories in Table 5 on page 42 can perform the attack. The automatically discovered attack violates both the security requirement \(\phi\) and the original SRP security goal which claimed, “The only possible attack against the protocol would be if nodes colluded during the two phases of a single route discovery” [23]. The NDA attack against SRP illustrated by our automated process uses no collusion and is performed by a single adversary. This attack was previously indicated by Buttyán and Vajda's simulatability model [21] and subsequently investigated and discovered through manual visual inspection. This research demonstrates the ability to discover the attack using automated model checking.

In addition to demonstrating the automated attack analysis process, we highlight the differences between a simulator and an exhaustive model checker, since SPIN can be used as either a simulator or as an exhaustive analyzer.

**SPIN simulation.** When using SPIN as a simulator, the outcome is limited by the initial seed value, so simulator outcomes do not guarantee that an attack does not exist. The seed
determines the process interleaving at each step. Since each node is a concurrent process and can operate independently, the seed determines whether a node receives or drops any incoming message.

Using SPIN in this fashion allows the possibility for any path to be exercised during the route discovery round, but does not exhaustively analyze all paths. For instance, the automated analysis over the NDA against the forward routing process should detect an error where SRP accepts a corrupt route, since the path check indicates the routing links that do not exist. SPIN simulations found non-attacked valid routes for the paths 0-4-3 and 0-1-3. After performing 300 simulations using different seed values, we did not find a case where SRP accepts a corrupt route. Seed dependent simulation may therefore lead us to believe that SRP is secure against this attacker, even when the SRP NDA attack indeed exists.

SPIN also allows interactive simulations. Performing an interactive SPIN simulation allows the analyst to choose the outcome for all non-deterministic decision points, showing that the SRP NDA is possible. While interactive simulation requiring human intervention for each decision point can identify attacks, it is a grueling non-exhaustive process.

**SPIN exhaustive analysis.** The power in our analysis approach lies in SPIN’s exhaustive verification method. SPIN can automatically check all possible message orderings until it finds an attack according to $\varphi$ or exhaustively determines that no attacks exist. The SPIN verifier automatically generates a trail file that can be read into the SPIN simulator to view the successful attack, without having to guess the seed to simulate the attack or interactively walk through all the simulation choices. As we have shown, evaluating SRP against the NDA using SPIN’s exhaustive verifier in the automated analysis process identifies the corrupted path 0-1-4-3, which SRP inappropriately accepts as a valid path.

### 7.2.3 Attacking Ariadne

Static analysis guided the Ariadne attacker development model in Section 6.5.3, indicating protocol messages may reveal information that allows a malicious insider (attacker category II) and a malicious outsider (attacker category I) the ability to drop a node from the forward rreq during Ariadne’s route discovery process.

**Ariadne NDA.** Attack cases for the NDA are obtained by running the automated analysis procedure against the Ariadne protocol and the category II attacker. Figure 58 illustrates an attacker case extracted from the Reporting Engine output file.
Processing case: ari_nda-rreq_5node_case-611.pml
Initiator Node: 0
Target Node: 3
Attacker Node: 4
Corrupted Path: 0 - 1 - 4 - 3
The topology is:

net_con[0].to[0] = 0 net_con[0].to[1] = 1 net_con[0].to[2] = 0 net_con[0].to[3] = 0 net_con[0].to[4] = 1

Figure 58. Automated Ariadne NDA Case

Viewing the automatically generated SPIN trail file for case 611 provides a message sequence leading to the Ariadne NDA. SPIN command line tools textually provide the messages leading to the attack. SPIN also can load the trail file graphically to view the attack sequence. For example, Figure 59 shows the ending of the SPIN attack sequence. The complete Ariadne NDA sequence generated by the trail file is given in Appendix B.

Figure 59. Ariadne Automated NDA Sequence

Following the generated attack sequence above and the Ariadne model message structure, defined in Section 6.4.3 on page 65, we re-illustrate the attack sequence and associated network configuration in Figure 60.
1-1 attacker, where the attacker has one compromised key loaded onto one attacker node, reported using visual inspection techniques. This research provides the ability to automatically can actively remove a node from the route during route discovery. This attack was previously process using SPIN discovered and identified an attack scenario where a single malicious insider corresponds to an active 1-1 attacker and the automated model checking cannot remove a node during the route discovery process. We show how an attacker using a insider, all the insider attacker categories listed in Table 5 on page 42 (categories II, IV, VI-IX) able to relay the return

crypt path in

\[ \phi \]

can be computed the matching hash value, and transmitted the corrupted path in

In order to actively remove a node from the accumulated path during the \textit{rreq}, the attacker requires the ability to generate the appropriate hash value, based on capturing a hash from an upstream node in the path. The attacker must be able to relay the return \textit{rrep} to any upstream neighbor explicitly listed in the unicast \textit{rrep}.

Since the category II attacker successfully breaks Ariadne with the NDA as a single malicious insider, all the insider attacker categories listed in Table 5 on page 42 (categories II, IV, VI-IX) can perform the attack. The automatically discovered Ariadne NDA violates both the security property \( \varphi \) and an original Ariadne security claim. In [24], the Ariadne authors claim that an active 1-1 attacker, where the attacker has one compromised key loaded onto one attacker node, cannot remove a node during the route discovery process. We show how an attacker using a malicious insider corresponds to an active 1-1 attacker and the automated model checking process using SPIN discovered and identified an attack scenario where a single malicious insider can actively remove a node from the route during route discovery. This attack was previously indicated by Ács, Buttyán, and Vajda [45] using their simulatability method and subsequently reported using visual inspection techniques. This research provides the ability to automatically discover the attack through model checking techniques.

In addition to demonstrating the developed automated security analysis process, we utilize the Ariadne NDA scenario as described in this section and the attacker development from Section 6.5.3 to show how SPIN models can generate attacks based on captured information during route discovery by modeling attacker memory to capture hash values.

**Ariadne INDA.** Analysis over Ariadne against a single malicious outsider (attacker category I) discovers a similar attack, with the exception that the attacker node 4 does not add its own id to the accumulated path in \( msg_4 \). The resulting corrupt path is 0-1-3. The automatically discovered Ariadne INDA violates both the security property \( \varphi \) and another original Ariadne security claim. In [24], the Ariadne authors claim an active 0-X attacker, where the attacker has no comprised keys

\[
\begin{align*}
\text{msg}_1 &= (\text{rreq, 0, 3, id}, h_0, (1, (1))) \\
h_0 &= H[0, \text{initiator-target secret}] \\
\text{msg}_2 &= (\text{rreq, 0, 3, id}, h_0, (1), (\text{sig}_0)) \\
h_0 &= H[1, h_0] \\
\text{sig}_0 &= \text{SK}_0 \{\text{rreq, 0, 3, id}, h_0, (1)\} \\
\text{msg}_3 &= (\text{rreq, 0, 3, id}, h_0, (1, 2), (\text{sig}_0, \text{sig}_1)) \\
h_0 &= H[2, h_0] \\
\text{sig}_0 &= \text{SK}_0 \{\text{rreq, 0, 3, id}, h_0, (1, 2), (\text{sig}_0)\} \\
*\text{msg}_4 &= (\text{rreq, 0, 3, id}, h_0, (1, 4), (\text{sig}_0, \text{sig}_1)) \\
h_0 &= H[4, h_0] = H[4, H[1, h_1]] \\
\text{sig}_0 &= \text{SK}_0 \{\text{rreq, 0, 3, id}, h_0, (1, 4), (\text{sig}_0)\} \\
\text{msg}_5 &= (\text{rreq, 0, 3, 3, id}, h_0, (1, 4), (\text{sig}_0, \text{sig}_1)) \\
\text{sig}_0 &= \text{SK}_0 \{\text{rreq, 0, 3, id}, h_0, (1, 4), (\text{sig}_0, \text{sig}_1)\}
\end{align*}
\]

**Figure 60. Ariadne Attack Sequence**
and any number of attacker nodes, can only perform a wormhole attack or force the protocol to choose an attacker desired path by rushing, which occurs when the attacker node responds to the route discovery process faster than the protocol expects [118].

We model the INDA using a malicious outsider (attacker category I), which corresponds to an active 0-1 attacker. The automated security analysis process using SPIN discovered and identified an attack scenario where a single outsider with no cryptographic keying material can actively remove a node from the path during route discovery. This is the first attack to our knowledge that shows a malicious outsider can change the path in Ariadne route discovery messages. A successful attack with a category I attacker indicates that any attacker within the attacker classification from Table 5 on page 42 can drop a node from the Ariadne route discovery process.

7.2.4 Attacking endairA

Static analysis guided the endairA attacker development model in Section 6.5.4, indicating that two colluding nodes may share cryptographic keys to generate the appropriate \textit{rrep} signatures in order to match NDA attempts against the forward endairA route discovery process. Attack cases for the 2-node NDA are obtained by running the automated analysis procedure against the endairA protocol and the category VII attacker. Figure 61 illustrates an attacker case extracted from the Reporting Engine output file.

<table>
<thead>
<tr>
<th>Processing case: end_nda-2_5node_case-250.pml</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiator Node: 0</td>
</tr>
<tr>
<td>Target Node: 2</td>
</tr>
<tr>
<td>Attacker Node: 4</td>
</tr>
<tr>
<td>Corrupted Path: 0 - 3 - 4 - 2</td>
</tr>
<tr>
<td>The topology:</td>
</tr>
<tr>
<td>\text{net_con}[0].\text{to}[0] = 0</td>
</tr>
<tr>
<td>\text{net_con}[1].\text{to}[0] = 0</td>
</tr>
<tr>
<td>\text{net_con}[2].\text{to}[0] = 0</td>
</tr>
<tr>
<td>\text{net_con}[3].\text{to}[0] = 1</td>
</tr>
<tr>
<td>\text{net_con}[4].\text{to}[0] = 1</td>
</tr>
</tbody>
</table>

Figure 61. Automated endairA NDA Case

Viewing the automatically generated SPIN trail file for case 250 provides a message sequence leading to the endairA NDA. SPIN command line tools textually provide the messages leading to the attack. SPIN also can load the trail file graphically to view the attack sequence. For example, Figure 62 shows the ending of the SPIN generated endairA attack sequence. The complete endairA NDA attack sequence generated by the trail file is given in Appendix B.
Following the generated attack sequence above and the endairA model message structure, defined in Section 6.4.4 on page 70, we re-illustrate the attack sequence and associated network configuration in Figure 63.

During the forward rreq, node 4 removes the intermediate node id from msg2 between itself and the colluding attacker node 3. During the rrep, node 4 signs for both itself and node 3, since the colluding nodes share their cryptographic keys. The initiator believes msg6 is from node 3,
since the signatures are correct and the initiator cannot identify that node 4 physically sent the message.

The automatically discovered endairA NDA violates both the specified security property \( \varphi \) and an original endairA security claim. In [45], the authors claim that endairA cannot be attack by colluding adversaries unless the adversaries are local neighbors to one another. In their model, local attacker neighbors are combined into a single adversary. The demonstrated colluding attacker shows that two colluding nodes may attack endairA without being local neighbors. To our knowledge, this is the first known attack that shows two colluding malicious insiders can actively corrupt the path in endairA route discovery messages. A successful attack using a category VII attacker indicates any multiple intruder classification (categories VII – IX) from Table 5 on page 42 can drop a node from the endairA route discovery process.

In addition to demonstrating the automated security analysis process, we utilize the endairA NDA attack scenario and the attacker development from Section 6.5.4 to show how SPIN models can generate attacks based on colluding attackers. In the model, an attacker node is able to discern if a colluding attacker has added its id to the route discovery path and it can react accordingly to initiate the attack.

7.3 Computational Requirements

Computational requirements are a limiting factor facing the model checking paradigm. In [110,109,95], research using SPIN to analyze loop freedom in MANET protocols focused respectively on three, four, and five node networks. In these studies, the authors were limited by state-space explosion related to modeling a protocol's ability to maintain proper routing operation as the network adapted due to mobility.

We focus on exhaustively evaluating static network topologies, since link failures due to mobility are not malicious. Following this approach, the automated security analysis process identified route discovery attacks against each analyzed source routing protocol using the network size \( N = 5 \). The computation requirements for the five-node models relate directly to the number of states that must be analyzed and the memory requirements to store each state.

The SPIN model checking environment performs a global state-space search, constructed by the combination of the independent system process components [63,107]. For instance, the FSA for each process type, as shown in Appendix A, represent the local states for an individual node of that process type. The global state-space is the combination of the active independent asynchronous node processes. Each global state is stored in a state vector, comprised over the global variables, channel contents, and the program counters and local variables for each process.

In order to construct the complete global system state required for the system models, we calculate the Cartesian product for the system's independent processes according to Equation 5,
where \( n \) represents the number of non-malicious nodes and \( m \) represents the number of malicious nodes.

**Total States** = (# of wireless medium states) \( n \) X (# of non-malicious states) \( n \) X (# of malicious states) \( m \) \hspace{1cm} (5)

The Cartesian product represents the worst case required to build the complete global system state representation. The worst-case total system states and respective memory requirements are listed in Table 8. The total states are determined according to Equation 5. For each model, the number of states required to control the wireless communication is 17.

### Table 8. Global System Requirements – Worst Case

<table>
<thead>
<tr>
<th>Model</th>
<th>Non-malicious local states</th>
<th>Non-malicious nodes</th>
<th>Malicious local states</th>
<th>Malicious nodes</th>
<th>Total states</th>
<th>State size</th>
<th>Total memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRP NDA</td>
<td>28</td>
<td>4</td>
<td>12</td>
<td>1</td>
<td>( \sim 1.25 \times 10^8 )</td>
<td>432 bytes</td>
<td>\sim 50 GB</td>
</tr>
<tr>
<td>Ariadne NDA</td>
<td>32</td>
<td>4</td>
<td>25</td>
<td>1</td>
<td>( \sim 4.46 \times 10^8 )</td>
<td>496 bytes</td>
<td>\sim 206 GB</td>
</tr>
<tr>
<td>Ariadne INDA</td>
<td>32</td>
<td>4</td>
<td>25</td>
<td>1</td>
<td>( \sim 4.46 \times 10^8 )</td>
<td>500 bytes</td>
<td>\sim 208 GB</td>
</tr>
<tr>
<td>endairA NDA</td>
<td>34</td>
<td>3</td>
<td>17</td>
<td>2</td>
<td>( \sim 1.93 \times 10^8 )</td>
<td>688 bytes</td>
<td>\sim 124 GB</td>
</tr>
</tbody>
</table>

Fortunately, SPIN’s on-the-fly model checking environment does not require the worst-case Cartesian product global state representation. SPIN’s reliance on linear temporal logic to build an event path towards a failure allows the states to be built on-the-fly, building states only when required along the current event path. SPIN takes advantage of the fact that although we are modeling distributed nodes that operate asynchronously, a node’s current state may be dependent on the state of the other system nodes. For instance, an intermediate node only interacts with a \texttt{rreq} message if the initiator has processed a request and the intermediate node has received the request from the wireless medium server. Building the states on-the-fly allows SPIN to analyze a system without building the worst-case state representation prior to analysis. Building the states as needed generally decreases the running time and storage requirements, since once a failure is found the analysis creates a trail file and terminates before creating all possible system states.

The state space can also be reduced through \textit{atomicity}, as described in 6.1 on page 50. By making executable Promela statements \textit{atomic}, system blocks execute without being interrupted to interleave executable statements from other concurrent processes. For instance, once a node receives a message, the message processing is performed atomically, without interruption, and sent to the wireless medium server destined for the next hop. Two nodes receiving the same message may still process the message in any order, since SPIN non-deterministically selects which node processes the pending message first. However, once a node is chosen to execute an atomic block, the local state of all other nodes is fixed until the atomic block completes or

95
deadlocks. Completeness is attained by SPIN’s exhaustive analysis searching all possible message processing orderings between the distributed nodes.

SPIN’s runtime state and memory requirements (in MegaBytes - MB) are thus much lower than the worst-case memory requirements from Table 8. The specific memory requirements are dependent on the network topology, or case number. Table 9 provides an example of the state and memory requirements, as reported by SPIN output. The runtime memory requirements are determined by: stored number of states \( X \) (state size + overhead). The relatively low memory requirements indicate the Promela models may be extendible to larger network topologies and model abstractions may be altered to capture more detailed protocol interactions.

Table 9. SPIN Runtime Requirements

<table>
<thead>
<tr>
<th>Model</th>
<th>Case #</th>
<th>Total unique states</th>
<th>Memory requirements (no compression)</th>
<th>Memory requirements (with compression)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRP NDA</td>
<td>627</td>
<td>20,858</td>
<td>9.261 MB</td>
<td>.902 MB</td>
</tr>
<tr>
<td>Ariadne NDA</td>
<td>615</td>
<td>2,081</td>
<td>1.057 MB</td>
<td>.281 MB</td>
</tr>
<tr>
<td>Ariadne INDA</td>
<td>1001</td>
<td>21,088</td>
<td>10.797 MB</td>
<td>.896 MB</td>
</tr>
<tr>
<td>endairA NDA</td>
<td>250</td>
<td>41,105</td>
<td>28.773 MB</td>
<td>1.510 MB</td>
</tr>
</tbody>
</table>

7.4 Chapter Summary

In this chapter we exercised the modeling approach developed in Chapter 6. In Section 7.1 we define the analysis criteria \( \varphi \) and discuss how the analysis criteria is implemented to check each returned route against the given network topology. In Section 7.2 we showed how the exhaustive analysis process automatically detected routing corruption attacks against three secure ad hoc source routing protocols: SRP, Ariadne, and endairA. The analysis process automatically detects a previously known attack against SRP and Ariadne. The automated security analysis process also discovers two new, previously unreported attacks. An against Ariadne by a malicious outsider and a colluding attack against endairA. Section 7.3 contains computational requirements encountered by the developed SPIN analysis models.
CHAPTER 8
CONCLUSIONS

Mobile ad hoc networks present an interesting computing paradigm, allowing rapid deployment of network services where fixed infrastructure may not be available. MANETs provide an attractive solution for disaster response scenarios, wildfire crews, and military applications. The core to MANET operation is the underlying routing protocol, allowing nodes to communicate with nodes outside their local 1-hop transmission distance. These protocols must provide security to the routing process. That is, the route discovery process should not be vulnerable to malicious message corruption that results in returned routes that do not exist within the current network topology.

In this thesis we show that a comprehensive security analysis framework advances the understanding of MANET routing protocol vulnerabilities. In Table 4 on page 12 we provided a comprehensive security evaluation framework that proposed combining visual inspection with formal methods to evaluate security properties during MANET route discovery. While visual inspection leverages human intuition to find attacks, it cannot provide exhaustive analysis to determine if an attack exists. Formal methods approaches thus far have not been widely adapted to evaluate MANET route security properties and did not provide automation. To provide a comprehensive security analysis capability, this thesis follows the formal methods approach by developing an automated security analysis procedure based on the model checking paradigm.

In the course of this research, we use the automated analysis process to discover two novel attacks against the secure routing process. A previously unreported attack against Ariadne using a malicious outsider and an attack against endairA using two colluding nodes.

8.1 The Problem

Initial research into MANET routing revealed that MANET security analysis generally follows unstructured, non-exhaustive, manual evaluation approaches. Informal security analysis approaches have led to an iterative cycle of research claiming secure routing protocols, followed by research offering attacks against such protocols. Many attacks are found through human intuition and visual inspection analysis that uncover straightforward flaws that were simply overlooked. However, visual inspection cannot exhaustively determine if an attack does not exist. The iterative cycle of protocol development and attack discovery commonly becomes confrontational, with claims about differences in environmental variables or assumptions, and claims that the security protocol was evaluated against an adversary that it was not intended to defend against. A secure routing protocol cannot be considered secure if the discovered routes
reflect paths that do not exist, regardless of the operational environment or the attacker assumptions.

### 8.2 Primary Contributions

This research provides a formalized security analysis process to evaluate MANET routing security. Combining an automated security analysis process with an adaptive threat model allows us to comprehensively determine the conditions under which a protocol fails. We follow the advice of Dijkstra [119], “One can never guarantee that a proof is correct, the best one can say: ‘I have not discovered any mistakes’ “. The research in this thesis does not aim to search for and claim unconditional protocol security, but provides a formal automated process to identify security flaws in MANET routing protocols. The automated security analysis process and the adaptive threat model aim to search for attacks according to Dijkstra’s viewpoint.

The adaptive attacker model developed in Chapter 5, enables us to change an attacker’s communication capability, whether an attacker is an insider or outsider, and whether an attacker operates individually or colludes with cooperating malicious nodes. The adaptive attacker classification provides the ability to determine the minimum attacker capabilities required to break a routing protocol, as opposed to claiming security based on a study’s stated assumptions.

The automated security analysis process developed in Chapter 7 provides a formal automated process that exhaustively evaluates all possible network configurations for a network size $N$ against the stated security goal $\phi$. We took a well known formal methods approach using model checking to evaluate security properties in authentication protocols and adapted it to support security analysis in the MANET routing paradigm.

The formal analysis models are developed in the Promela modeling language and evaluated using the SPIN exhaustive model checking environment. Through case studies, the automated analysis models automatically discover known attacks against SRP and Ariadne, and identify previously unreported attacks against both Ariadne and endairA. During model demonstration, we discover a novel attack against Ariadne in which a malicious outsider can change the path embedded in Ariadne routing messages. We also discovered a novel attack on endairA, in which two nodes collude to attack the path in endairA routing messages. In both attacks, the altered path tricks the initiator into using a path that does not exist.

The research provided in this thesis is the first known automated model checking approach to evaluate route security in on-demand source routing protocols.

### 8.3 Future Work

The research in this thesis provides a foundation for continued work. The relatively low runtime memory requirements identified in Section 7.3 indicate that the SPIN models may be
extendible to larger network topologies and suggest that modeled protocols may be altered to capture more precise protocol interactions.

Models that incorporate protocol interactions at a lower abstraction level by including more detail may enable future research into automated code generation for MANET routing protocol development. This capability would allow strict software engineering practices to specify and develop the routing protocol operations based on its intended goals. The protocols could be initially developed in a model checking environment based solely on specifications and requirements. Once the operation and security properties are validated in the model checking environment, automated code generation may be able to produce executable code for real-world MANET implementations.

Another area of research sparked by our foundational work is to develop an associated formal process algebra to correspond to the SPIN/Promela environment. A formal process algebra would allow us to mathematically expand results obtained by the finite SPIN models to infinite systems or systems too large to model within resource constraints.
APPENDIX A

SPIN GENERATED FINITE STATE AUTOMATA

Figure 64. Wireless Medium Server FSA
Figure 65. DSR Non-malicious Node FSA
Figure 66. SRP Non-malicious Node FSA
Figure 67. Ariadne Non-malicious Node FSA
Figure 68. endairA Non-malicious Node FSA
Figure 69. INA FSA

Figure 70. SRP NDA FSA
Figure 71. Ariadne NDA FSA
Figure 72. Ariadne INDA FSA
Figure 73. endairA NDA FSA
APPENDIX B

SPIN TRAIL FILES

Figure 74. SRP NDA Trail File
Figure 74 Continued
Figure 75. Ariadne NDA Trail File
Figure 75 Continued
Figure 76. endairA NDA Trail File
Figure 76 Continued
APPENDIX C

COPYRIGHT PERMISSION FORMS

COPYRIGHT PERMISSION FORM

Dear Jacqueline Hansson (IEEE Intellectual Property Rights Office):

I am completing dissertation at Florida State University entitled "Formal Security Evaluation of Ad Hoc Routing Protocols." I would like your permission to reprint in my dissertation excerpts from the following work in which I am the original author:


The requested permission extends to any future revisions and editions of my dissertation, including nonexclusive world rights in all languages. These rights will in no way restrict republication of the material in any other form by you or by others authorized by you. This authorization is extended to University Microfilm Inc. / ProQuest Information and Learning, Ann Arbor, Michigan, for the purpose of reproducing and distributing copies of this dissertation. Your signing of this letter will also confirm that you own [or your Company owns] the copyright to the above described material.

If these arrangements meet with your approval, please sign and date this letter where indicated below and return it to me in the enclosed return envelope. Thank you very much.

Sincerely,

Todd R. Andel

PERMISSION GRANTED FOR THE USE REQUESTED ABOVE:

Jacqueline Hansson
IEEE Intellectual Property Rights Office
445 Hoes Lane
Piscataway, NJ 08855-1331 USA

Date: 6/22/07

RECEIVED
JUN 22 2007
COPYRIGHTS
REFERENCES


Todd R. Andel was born, raised, and attended high school in northern Wisconsin. In 1998, he graduated Summa Cum Laude with a Bachelor of Science in Computer Engineering from the University of Central Florida. He graduated with a Master of Science in Computer Engineering from the Air Force Institute of Technology in 2002.

In 2004, Todd entered the doctoral program in the Department of Computer Science at Florida State University. His research interests include wireless secure routing protocols, simulation, and formal methods. He is currently a member of the Institute of Electrical and Electronic Engineers (IEEE) and the Association of Computing Machinery (ACM).