Anonymous Routing in Mobile Ad Hoc Networks

Neer Roggel
Anonymous Routing in Mobile Ad Hoc Networks

Research Thesis

Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Science

Neer Roggel

Submitted to the Senate of the Technion — Israel Institute of Technology

NISAN, 5771 Haifa April, 2011
This Research Thesis was completed under the supervision of Roy Friedman in the Department of Computer Science

I thank everyone for everything. My beloved family for their loving support, my courageous friends for their encouragements, my patient advisor for his insightful perspectives and the many more who have taught me along the way and who shall, fittingly, remain anonymous.

Amir, Rachelly and Bar, for inspiring me to work hard through personal example, for ever providing me with unconditional security and for wishing me naught but happiness.

Daniel, for his heartening fortitude, for his hospitality and for good times aplenty. Karine, for her tenacity, candor and special audacity. Gal, for sharing his unique clarity of thought and good cheer. Orr, for his quiet chivalry and for introducing me to the Technion.

Roy, for getting me going, for vivid archery analogies, for a wonderful trip to Finland, for listening to endless internal debate, for helping me to structure my arguments, for ensuring all is kept in balance, for reading endless drafts, for offering his invaluable advice and for maintaining calm with a smile and gentle kindness (and for added financial assistance).

The generous financial help of the Technion is gratefully acknowledged.
# Contents

Abstract 1

List of Symbols 3

1 Introduction 5

2 Models 9

2.1 Items of interest ................................................. 9
2.2 System model .................................................... 9
2.3 Network model .................................................. 10
2.4 Threat model .................................................... 11
    2.4.1 Attacker capabilities ..................................... 11
    2.4.2 Modi operandi ............................................. 15
2.5 Adversary model .............................................. 18
2.6 Trust model ................................................... 19
2.7 Motion model .................................................. 19

3 Goals 21

3.1 Anonymity (and friends) ..................................... 21
    3.1.1 Unobservability .......................................... 22
    3.1.2 Immeasurability .......................................... 22
    3.1.3 Anonymity ................................................. 23
    3.1.4 Comparison ............................................... 24
3.2 Communication (and routing) ................................. 24
    3.2.1 Procedures ............................................... 25
    3.2.2 Invocations ............................................... 25
    3.2.3 Identities ............................................... 26
3.3 Performance ................................................... 28
3.3.1 Complexity .................................................. 28  
3.3.2 Cryptography ................................................. 28  
3.3.3 Heterogeneity ............................................... 28  
3.3.4 Implementability .......................................... 28  
3.3.5 Robustness ................................................ 28  
3.3.6 Scalability ............................................... 28  
3.4 Problem statement .......................................... 29  

4 Anonymity-blásé systems ............................................. 31  
4.1 Flood Routing Algorithms .................................... 31  
4.1.1 Assumptions ................................................. 32  
4.1.2 Specification ............................................... 32  
4.2 Dynamic Source Routing (DSR) ................................ 35  
4.2.1 Assumptions ................................................. 35  
4.2.2 Specification ............................................... 35  

5 Analysis .......................................................... 37  
5.1 Observations .................................................. 37  
5.2 Limitations .................................................... 38  
5.3 Preliminaries ................................................... 39  

6 Mononymous routing ................................................ 41  
6.1 Agreements ...................................................... 42  
6.2 Mononymous FRAs ............................................ 42  
6.3 Mononymous DSR ............................................. 45  

7 Polyonymous routing ................................................. 47  
7.1 Agreements ...................................................... 48  
7.2 Polyonymous FRAs ............................................ 48  
7.3 Polyonymous DSR ............................................. 50  

8 Anonymity-aware systems ........................................... 51  
8.1 Mix Route Algorithm .......................................... 52  
8.1.1 Assumptions ................................................. 52  
8.1.2 Specification ............................................... 53  
8.2 Secure Distributed Anonymous Routing (SDAR) .......... 54
Contents (continued)

8.2.1 Assumptions ......................................................... 54
8.2.2 Specification ......................................................... 55
8.3 Anonymous Routing Protocol for MANETs (ARM) ......................... 56
  8.3.1 Assumptions ......................................................... 56
  8.3.2 Specification ......................................................... 57

9 Comparison 59

10 Related work 63
  10.1 Anonymity in wired networks ........................................ 63
  10.2 Anonymity in MANETs ............................................. 64

11 Conclusion 65
  11.1 Summary ........................................................... 65
  11.2 Discussion .......................................................... 66
  11.3 Contributions ....................................................... 67
  11.4 Insights ............................................................. 68
  11.5 Open issues ......................................................... 68

A Optional features 71

B Procedures of DSR 75

C Wormhole routing 81
  C.1 Assumptions ......................................................... 81
  C.2 Specification ......................................................... 82

References 82

Hebrew Abstract i
List of Figures

1.1 Chapter flow with inter- and intra-chapter dependencies . . . . . . . . . . . 7
2.1 A sample classification of items of interest . . . . . . . . . . . . . . . . . . 10
2.2 Relationships between adversarial capabilities . . . . . . . . . . . . . . . . 12
3.1 Network-Layer-Send() Procedure . . . . . . . . . . . . . . . . . . . . . . . . 25
3.2 Network-Layer-Recv() Procedure . . . . . . . . . . . . . . . . . . . . . . . . 25
4.1 Route-On-Send() Procedure for FRAs . . . . . . . . . . . . . . . . . . . . . . 33
4.2 Filter() Procedure for FRAs . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
4.3 For-Me() Procedure for FRAs . . . . . . . . . . . . . . . . . . . . . . . . . . 33
4.4 Accept() Procedure for FRAs . . . . . . . . . . . . . . . . . . . . . . . . . . 33
4.5 Route-On-Recv() Procedure for FRAs . . . . . . . . . . . . . . . . . . . . . 34
4.6 Encode-Destination-Indicator() Procedure for FRAs . . . . . . . . . . . . . 34
4.7 Decode-Destination-Indicator() Procedure for FRAs . . . . . . . . . . . . . 34
B.1 Route-On-Send() Procedure for DSR . . . . . . . . . . . . . . . . . . . . . . 76
B.2 Filter() Procedure for DSR . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
B.3 For-Me() Procedure for DSR . . . . . . . . . . . . . . . . . . . . . . . . . . 77
B.4 Accept() Procedure for DSR . . . . . . . . . . . . . . . . . . . . . . . . . . 77
B.5 Route-On-Recv() Procedure for DSR . . . . . . . . . . . . . . . . . . . . . 78
B.6 Encode-Destination-Indicator() Procedure for DSR . . . . . . . . . . . . . 78
B.7 Decode-Destination-Indicator() Procedure for DSR . . . . . . . . . . . . . 78
B.8 Encode-Router-Indicator() Procedure for DSR . . . . . . . . . . . . . . . 79
B.9 Decode-Router-Indicator() Procedure for DSR . . . . . . . . . . . . . . . 79
C.1 Route-On-Send() Procedure for WHR . . . . . . . . . . . . . . . . . . . . . . 83
C.2 Filter() Procedure for WHR . . . . . . . . . . . . . . . . . . . . . . . . . . . 83
C.3 For-Me() Procedure for WHR . . . . . . . . . . . . . . . . . . . . . . . . . . 83
C.4 Accept() Procedure for WHR . . . . . . . . . . . . . . . . . . . . . . . . . . 84
C.5 Route-On-Recv() Procedure for WHR . . . . . . . . . . . . . . . . . . . . . 84
Abstract

A wireless, mobile, ad hoc network (MANET) is a network in which mobile nodes do not rely on the existence of fixed infrastructure mediation devices, but rather communicate with one another directly. Under certain scenarios, parties in a MANET may wish to remain unidentified, in order to forestall retaliation by an attacker. In the course of this work, we study mechanisms for anonymous routing in MANETs. As our first main contribution, we construct a simple framework for formal reasoning about anonymous routing in MANETs, within which we prove our results. We describe routing and identities, explore threats to anonymity in MANETs, derive a suitable adversary model and use it to define several notions of anonymity in a formal yet intuitive manner. As our second main contribution, we prove that use of unbiased identity agreements, in which addresses are equally likely to be assigned to different nodes, lets existing routing algorithms achieve anonymity. As our third main contribution, we prove that use of multiple identities per node can increase information hiding. Throughout, we avoid cryptography and defend against a weakened adversary. Finally, we survey existing anonymization schemes and compare approaches.
## List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I$</td>
<td>the set of all items of interest</td>
</tr>
<tr>
<td>$UI$</td>
<td>the set of unobservables ($\subseteq I$)</td>
</tr>
<tr>
<td>$OI$</td>
<td>the set of observables ($= I \setminus UI$)</td>
</tr>
<tr>
<td>$COI$</td>
<td>the set of controllable observables ($\subseteq OI$)</td>
</tr>
<tr>
<td>$UOI$</td>
<td>the set of uncontrollable observables ($= OI \setminus COI$)</td>
</tr>
<tr>
<td>$D$</td>
<td>the set of all devices</td>
</tr>
<tr>
<td>$P$</td>
<td>a set of processors ($\subseteq D$)</td>
</tr>
<tr>
<td>$p_i, p_j$</td>
<td>member processors ($\in P$)</td>
</tr>
<tr>
<td>$n$</td>
<td>a processor count ($\in \mathbb{N}$)</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>an execution</td>
</tr>
<tr>
<td>$A$</td>
<td>an adversary</td>
</tr>
<tr>
<td>$\text{view}^A_{\alpha}$</td>
<td>$A$’s view of $\alpha$</td>
</tr>
<tr>
<td>$C_A$</td>
<td>“colors” discernible by $A$</td>
</tr>
<tr>
<td>$c$</td>
<td>a discernible “color” ($\in C_A$)</td>
</tr>
<tr>
<td>$X$</td>
<td>an input space</td>
</tr>
<tr>
<td>$x$</td>
<td>an input ($\in X$)</td>
</tr>
<tr>
<td>$\text{ALG}$</td>
<td>a distributed algorithm</td>
</tr>
<tr>
<td>$E_{x \rightarrow c}^{\text{ALG}}$</td>
<td>an event: $\text{ALG}$ produces $c$ when fed $x$</td>
</tr>
<tr>
<td>$\sim_{\text{ALG}}$</td>
<td>input indistinguishability induced by $\text{ALG}$</td>
</tr>
<tr>
<td>$\simeq$</td>
<td>an isomorphism relation between inputs</td>
</tr>
<tr>
<td>$\leq$</td>
<td>information hiding order on algorithms</td>
</tr>
<tr>
<td>$G$</td>
<td>a comm. multigraph</td>
</tr>
<tr>
<td>$t$</td>
<td>a private target value</td>
</tr>
<tr>
<td>$T_i$</td>
<td>the set of target values at $p_i$</td>
</tr>
<tr>
<td>$T$</td>
<td>the set of target values at all nodes</td>
</tr>
<tr>
<td>$\tau$</td>
<td>an arc labeling, giving target values</td>
</tr>
<tr>
<td>$A$</td>
<td>a global address space</td>
</tr>
<tr>
<td>$a$</td>
<td>an address ($\in A$)</td>
</tr>
<tr>
<td>Symbol</td>
<td>Meaning</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>an agreement on addresses</td>
</tr>
<tr>
<td>$K(G)$</td>
<td>the set of agreements over $G$</td>
</tr>
<tr>
<td>$GenID$</td>
<td>a distributed agreement algorithm</td>
</tr>
<tr>
<td>$E_{GenID}^G$</td>
<td>an event: $GenID$ labels $G$ according to $\kappa$</td>
</tr>
<tr>
<td>$E_{E \rightarrow a}^G$</td>
<td>an event: $GenID$ labels the arc $e$ with $a$</td>
</tr>
<tr>
<td>$f_G$</td>
<td>a mapping of agreements to observed “colors”</td>
</tr>
<tr>
<td>$K_{G \rightarrow c}$</td>
<td>the set of agreements for which $f_G$ produces $c$</td>
</tr>
<tr>
<td>$\mathcal{B}(G)$</td>
<td>the set of “bundles” of atomically labeled arcs in $G$</td>
</tr>
<tr>
<td>$\simeq_{GenID}$</td>
<td>isomorphism under $GenID$ with respect to “bundles”</td>
</tr>
<tr>
<td>$\Pi_{G_1, G_2}^{GenID}$</td>
<td>the set of all such isomorphisms going from $G_1$ to $G_2$</td>
</tr>
<tr>
<td>$\pi$</td>
<td>one such isomorphism going from $G_1$ to $G_2$ ($\in \Pi_{G_1, G_2}^{GenID}$)</td>
</tr>
<tr>
<td>$h$</td>
<td>a hash function</td>
</tr>
<tr>
<td>$Q$</td>
<td>a probability distribution over the address space</td>
</tr>
<tr>
<td>$\mathcal{P}(X)$</td>
<td>the set of all multisets overlying the set $X$</td>
</tr>
<tr>
<td>$X/\equiv$</td>
<td>partition of $X$ induced by the equivalence $\equiv$</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

A wireless, mobile, ad hoc network (MANET) is a network in which communicating mobile nodes do not rely on the existence of fixed infrastructure mediation devices, but rather communicate with one another directly. Wireless MANETs are often found deployed where unfavorable conditions deter the deployment of a fixed wire network, where anonymity may be highly desirable for participating nodes. Consider a scenario in which members of an underground movement wish to share news about the crimes of an oppressive regime. Can an attacker detect who is producing the information and who is consuming it? Can an attacker ascertain other relationships between participating nodes and consequently punish them? Alternatively, consider a mobile combat unit operating inside enemy territory. Given that an enemy has set up a sensor network to eavesdrop on all communications, does the enemy know which node serves as a central leader or where forces are concentrated?

On account of quandaries such as these, anonymity in MANETs has become a subject of research in recent years, with several anonymization schemes proposed and analyzed. These include refurbished wired network schemes as well as specially tailored new ones. However, most existing schemes require use of heavy cryptography, pre-existing trust among nodes or dissemination of complete knowledge of network topology. These requirements are not aligned with typical MANET constraints, whereby nodes need to conserve energy, do not necessarily know each other in advance and may often move about, respectively. Moreover, adversaries encountered in wireless MANETs are quite different from those encountered in fixed networks, due to increased node vulnerability and the nature of the wireless medium. Hence, anonymity in MANETs requires rethinking. In this work, we examine it anew.

While anonymity has had many a definition in the literature over the years, it remains clear that the problem is a multifaceted one, in that many system components can affect anonymity, both separately and in combination. In this work, we focus on one particular aspect of anonymity – that of anonymous routing. The problem of anonymous routing poses an apparent contradiction between two conflicting requirements that must be reconciled. On the one hand, the network is required to deliver messages at their destinations and at their destinations only; symmetry must be broken. On the other hand, the network is required to do so without identifying these same destinations; symmetry must be maintained.

In order to satisfy the first requirement, one may employ the abstraction of an identity. An identity is used to refer to a node or an entity in the network. By their very definition, identities allow differentiation of entities in the system. However, as an unfortunate result, a divulged identity can lead back to and bring undesirable reprimand to its owner.
In order to satisfy the second requirement, current approaches to anonymous routing in MANETs sometimes utilize a supplementary layer of cryptographic identities that hides existing identities; with their identities hidden, communicating nodes remain unidentified. In the course of this work, we propose and study an alternative approach. We observe that the basic assumption and rationale underpinning the above logic – that logical identities reveal “true” identities – does not apply to MANETs. On the Internet, a logical identity is presumed to expose its owner, as per the ability to obtain customer records from service providers. Within an enterprise network, an administrator may provide the required details. However, in MANETs, and in general, logical identities are neither intrinsic to devices nor necessarily correlated with them. Hence, when designing a system for anonymous routing, logical identities are not required to be hidden – only their association with their owners is. Given the above realization, instead of restoring symmetry with cryptography, we propose to achieve anonymity by utilizing symmetric identity agreements, at the onset. As it were, we propose to settle the contradiction by breaking symmetry in a symmetric way.

As our first main contribution, perhaps being the most foundational one, we construct the necessary machinery for formal reasoning about anonymous routing in MANETs. In a nutshell, we develop a simple framework or “language” that enables us to formally express statements about anonymity, as well as to compare the anonymity of different algorithms. With our new framework in place, we proceed to state and prove our obtained results. As our second main contribution, we propose that existing algorithms simply make use of identity agreements in which addresses are equally likely to be assigned to different nodes. Consequently, an adversary observing an execution of the system is unconditionally unable to determine which nodes are using which addresses. As our third main contribution, we propose the use of multiple identities per node to further increase information hiding.

Our work is structured as follows. In chapter 2, we present the models with which we investigate anonymity in MANETs. In particular, we define MANETs, explore threats to anonymity in MANETs and derive a suitable adversary model. In chapter 3, we present our goals when designing a system for anonymous routing. In particular, we formally define anonymity in terms of our adversary model, as well as routing algorithms and their use of identities. In chapter 4, we present two existing routing algorithms, designed without anonymity in mind, upon which we demonstrate our approach. In chapters 5, 6 and 7, we formally analyze their information hiding properties under various identity agreements. In chapter 5, we present the observations that led to our approach, the limitations of our analysis and preliminary definitions. In chapter 6, the two algorithms are analyzed under the constraint of a single identity per node. In particular, we prove that under an unbiased identity agreement, anonymity is achieved. In chapter 7, the analysis is extended to cases of multiple identities per node. Then, in chapter 8, we present three existing anonymization schemes for MANETs. In chapter 9, we compare their approach with ours. In chapter 10, we present additional related work. In chapter 11, we present concluding remarks. A more detailed chapter flow with inter- and intra-chapter dependencies is shown in Figure 1.1.

Throughout the work, we explicitly adopt simplifying assumptions in an effort to limit the scope of our analysis. Most notably, we cripple the adversary plus ignore spacetime, message contents and identity agreement implementation. Thus, though the principles at work remain sound, our results pertain to distilled scenarios only. In turn, the work is better classified as preliminary groundwork rather than comprehensive analysis. With our framework firmly in place, we defer analysis of more capable adversaries to future work.
Figure 1.1: Chapter flow with inter- and intra-chapter dependencies. Chapters flow from top to bottom. Subject matter appears in sequence and in correspondence with chapter sections (where applicable). An arrow from A to B indicates that B depends on A.

Items of Interest

- System Model
- Network Model
- Threat Model
- Adversary Model
- Trust Model
- Motion Model

Chapter 1: Introduction

Chapter 2: Models

Chapter 3: Goals

Chapter 4: Anonymity-blasé Systems

Chapter 5: Analysis

Chapter 6: Mononymous Routing

Chapter 7: Polyonymous Routing

Chapter 8: Anonymity-aware Systems

Chapter 9: Comparison

Chapter 10: Related Work

Chapter 11: Conclusion
Chapter 2

Models

In this chapter, we present our system model, network model, threat model, adversary model, trust model and motion model. It is within the framework of these models that we are able to present our claims and prove our results in the chapters that follow.

2.1 Items of interest

In the study of anonymity, system elements that might affect anonymity are referred to as items of interest [79]. In this work, we further subdivide these, as follows. Let \( I \) be the set of all such items. Let \( OI \subseteq I \) be the set of those potentially observed by an attacker (observables). Let \( UI = I \setminus OI \) be the rest (unobservables). Let \( COI \subseteq OI \) be the set of observables which can be algorithmically controlled. Let \( UOI = OI \setminus COI \) be the rest.

A sample classification of items of interest in MANETs is shown in Figure 2.1, which serves to provide a rough map of system elements that might affect anonymity, and that must be considered when designing an anonymous system. In the figure, \( UI \) includes items which cannot be seen and must be inferred, while \( COI \) and \( UOI \) include items which can be seen and from which inferences are made. A classification such as this one allows us to, on the one hand, place various aspects of system operation on the same footing and, on the other, focus on particular ones while ignoring others. In particular, this work primarily focuses on the relation of unobservable communication patterns to observable addresses.

2.2 System model

Let \( D \) be the set of all devices. A mobile ad hoc network, or MANET, is a packet switching radio network over a subset of \( D \). A device can simultaneously belong to more than one MANET. Different MANETs can cohabit the same region of spacetime and can partition and coalesce. A device that belongs to a MANET is also referred to as a node. In formal terms, we model nodes in MANETs as processors in message passing systems [5].

Let us consider a message passing system with \( n \in \mathbb{N} \) processors. Let \( P \subseteq D \) be the set of processors in the system, modeled as non-deterministic state machines. Occurrences in the system are modeled as events. In addition to local computation events and message
delivery events, we consider an emission event, denoted emit\((i, oi)\), which represents the emission of an observable \(oi \in OI\) by the processor \(p_i \in P\). In our model, every delivery event, del\((i, j, m)\), is accompanied by a corresponding emission event by \(p_i\), which contains the message \(m\), including information from any employed message delivery and processor identification mechanism. This captures the notion that a transmitted message is potentially observed by an eavesdropping attacker, and may contain node addresses. If message reception is also observable, del\((i, j, m)\) implies an analogous emission event by \(p_j\).

Let \(emissions_\alpha\) denote the set of observables emitted in an execution \(\alpha\) of a system. By convention, we assume that all observables generated in an execution are emitted. We refer to the sum total of emissions in a given execution as the transcript of the execution.

Without loss of generality, we assume that emission of controllable observable items is determined by the system’s initial configuration (which specifies the processors’ initial states and encodes the input) and by the processors’ non-determinism.

### 2.3 Network model

We model links between nodes in MANETs as arcs in a datalink layer topology digraph. A characteristic feature of MANETs is that the physical layer and (resulting) datalink layer topology graphs are not static, due to ongoing node motion and link interference.

We assume the use of a layer model like that of OSI [101]. We note that in such a model, anonymity may be compromised in any layer. In MANETs, the distinction between
datalink layer (e.g., MAC) addresses and network layer (e.g., IP) addresses is loose and somewhat arbitrary, because every node can serve as both a router and an endpoint.

Our work is not concerned with datalink layer addresses, because these do not directly indicate relationships between entities at the network level, assuming that no identity binding protocol (e.g., ARP [80]) is in place. In addition, we deliberately assume that upper layer data does not compromise anonymity\(^1\), and focus on providing network layer anonymity. Still, we sometimes refer to upper layer data that warrants protection of the identities of any devices involved in its handling as questionable content or data.

Lastly, we assume the following general communication strategy. Our work picks up anonymity at stages 1, 3 and 4, whereas stages 2 and 5 are outside the scope.

1. Joining a network.
2. Searching for and deciding upon a desired communication partner. If a suitable partner is not found, leaving the network or trying again. If it is found or is known to be on the network in advance, continuing to the next stage.
3. Attaining whatever information is necessary for the routing algorithms in use to bring about the desired communication, assuming that such information is not already present (possibly distributed in the network) in advance.
4. Invoking the routing algorithms to help establish a connection and maintain it while transferring data between the communicating partners.
5. Leaving the network once communication is complete.

### 2.4 Threat model

In this section, we explore threats to anonymity in MANETs, in an effort to derive an appropriate adversary model, to be presented in the next section. We consider technical adversary capabilities separately from modi operandi that utilize them. An adversary’s extent is defined in terms of OSI layers, time and geography – each capability is exercisable in a given set of layers, in a given spacetime region, at a possibly bounded rate.

#### 2.4.1 Attacker capabilities

We present adversarial capabilities in a loosely hierarchical order – capabilities presented first serve as building blocks for latter ones. Figure 2.2 summarizes the relationships between capabilities. Generally, except in the case of capabilities that require direct manipulation of the physical layer, an adversary may use commodity hardware.

\(^1\)Although sometimes true, this assumption is generally unjustified. We deliberately ignore message contents in order to separately study mechanisms for network layer anonymity and for message hiding.
Eavesdropping

An attacker with proper equipment is capable of intercepting layer 1 transmissions as they propagate in his vicinity. An attacker can expand his area of vicinity by employing more sensitive equipment. Less expensively, in the case of a broadcast medium, any node can easily eavesdrop on layer 2 transmissions by setting its NIC to “promiscuous mode”. The eavesdropping attacker learns the contents of messages as they traverse the medium.

In terms of extent, an attacker may be able to eavesdrop on all messages passed in the system. In this case, he is a global eavesdropper. Otherwise, an attacker may only be able to eavesdrop on a given subset of messages at a time. An example of this is when communication links are geographically widespread, but attacker motion patterns are limited and do not allow him to be omnipresent. In this case, we say that the attacker is a local eavesdropper, and his locality can then be further quantified.

It is natural to assume that an attacker knows the geographical location of a node that generates or receives eavesdropped data. However, in some cases, such as when working with a large aggregation of eavesdropped data, or when the eavesdropping equipment cannot accurately pinpoint the source or destination of a transmission, this assumption is not justified. Therefore, we refine the notion of an eavesdropping attacker and define an attacker’s spatial eavesdropping resolution in the next segment.

In wired and hybrid networks, an attacker can utilize lawful interception technologies available in the network equipment used by victims in order to perform eavesdropping [62]. However, in the case of MANETs, the data is not necessarily relinquished by the nodes or available at any central authority, making MANETs more resistant to this attack vector.

We assume that the act of eavesdropping is undetectable by the victims.

Figure 2.2: Relationships between adversarial capabilities.

An arrow from A to B indicates that A is facilitated by B.
Transmitter geolocation

An attacker can geographically pinpoint a transmitter, by using the above layer 1 equipment combined with positioning techniques, including signal triangulation, signal strength analysis [85][60] and active disruptions followed by handoff or route change detection.

An attacker with perfect spatial eavesdropping resolution can pinpoint the source or destination of any transmission and attribute it to a single device. An attacker with partial spatial eavesdropping resolution can only attribute the transmission to one of several devices. An attacker with zero spatial eavesdropping resolution knows only the contents of the eavesdropped messages. Interestingly, the use of omnidirectional antennae aids anonymity by potentially posing an upper bound on an attacker’s eavesdropping resolution.

On the Internet, an ISP can act as a “location service” that allows law enforcement agencies to geographically locate a node with a given IP address (for this reason, proxying anonymity solutions [35][1] prefer to select proxies residing under ISPs in different countries). However, in the case of MANETs, node locations are not fixed and more difficult to track, making MANETs more resistant to this attack vector. In the case of a MANET that coexists with a centrally managed network carrying location information, as can happen when MANET nodes are also part of a cellular network, an attacker can also extract location information from the network center (e.g., which handset was in which cell at what time [14][13]). In the case of a network of position-aware devices [68], such information might be voluntarily exchanged in the clear between the devices, be they compromised or not.

Device fingerprinting

When the information obtained using the layer 1 equipment above is not sufficient to yield a device’s location, it may still be sufficient in order to identify that the same device is transmitting at different times, using passive analysis. For instance, in an environment with many wireless devices that are stationary in relation to the attacker, the attacker can use observed signal strength as an indicator of device identity to realize that two observed packets originated from the same device [31].

An alternative approach is to use sophisticated surveillance equipment to extract physically identifying characteristics. This includes using cameras or satellite imagery to identify and track devices, or detecting unique RF signal artifacts that can identify WICs [81].

Device fingerprinting can also be achieved in higher layers. At the datalink and network layers, if devices use fixed datalink or network layer addresses, an eavesdropper can trivially detect that the same device is transmitting multiple times (assuming no spoofing occurs). Less trivially, an attacker can actively probe a host with ICMP or TCP packets to reveal the host’s operating system [4][97], observe a host’s TCP connections to reveal its unique internal clock skew [57], or actively engage the host in order to induce processor temperature changes that result in observable clock skew changes [71]. At the application layer, devices can be tracked by planting bugs [2], or by observing user identities sent in the clear.

We define an attacker’s fingerprinting resolution as the attacker’s ability to recognize the same device across time and distinguish it from other devices. In other words, fingerprinting resolution is an attacker’s ability to tell whether or not a device is different from all others it has encountered before, and if not, tell which one of the ones it has encountered it is. An
attacker with *perfect fingerprinting resolution* can detect the same device across time and can completely distinguish devices from one another. An attacker with *partial fingerprinting resolution* can only attribute a fingerprint to one of several devices\(^2\). An attacker with *zero fingerprinting resolution* cannot distinguish between different devices.

It is important to refine the above definition to account for contexts. For instance, one device may be distinguishable from another device in the context of eavesdropping on fixed logical addresses, but not in the context of geolocating device transmissions. We note that even an attacker with perfect fingerprinting resolution in different contexts does not necessarily possess the ability to correlate devices across contexts. For instance, a global eavesdropping attacker that can both perfectly resolve devices based on their use of fixed logical addresses and perfectly resolve devices based on their location as indicated by satellite imagery does not necessarily know the location of a specific device that uses a specific fixed logical address to access questionable content. For that, the eavesdropper might require the added capability of geolocating device transmissions.

**Traffic generation**

An attacker can gain physical access to the medium and start transmitting. In the case of a wireless medium, this type of access is very easy to come by, and an attacker can transmit from very far away by using a high gain antenna to boost his signal [30][18]. Traffic can be generated for the sake of affecting network load, thereby causing routing and timing changes, or for the sake of participating in a given network protocol.

**Man-in-the-middle**

An attacker can combine eavesdropping and traffic generation in order to intercept node transmissions and change them en route. In the physical layer, this can be done by jamming the original signal or replacing it with a stronger signal. In higher layers, this can be achieved by exploiting weaknesses in network protocols (e.g., ARP and DNS poisoning [70], TCP hijacking [53], rogue access points [48], et cetera).

A man-in-the-middle threatens anonymity in two ways. First, in some protocols, such as IKE [55], participant identities are normally hidden but are revealed to an active man in the middle [76]. Second, a man-in-the-middle attacker can logically compromise a node by injecting modified executables in response to a user initiated software download request [73]. A specific victim can be chosen based on results of traffic analysis or based on a priori knowledge. Once the victim has been compromised, it can be more easily located, or if it is a position-aware device, then its position is immediately revealed to the attacker.

**Logical compromise**

An attacker can exploit a vulnerability in a logically accessible node, in order to compromise that node. Once compromised, the attacker can perform any operation that can be

\(^2\)As an anecdote and an example of partial fingerprinting resolution outside the context of MANETs, consider that the author went to high school with a fellow student named Will Smith, who, given an identically named actor of Hollywood fame, is nearly impossible to locate in an online search.
performed by software on that node. This includes using the device as a bug, recording its communications and recording routing information, among others. We assume that node owners remain unaware of the compromise and of the attacker’s operations.

**Physical compromise**

An attacker can physically come into contact with an unattended node, perhaps during manufacture time, and replace its hardware or software with his own. Once compromised, the node lets the attacker perform any operation that can be performed by said hardware or software. We assume that a physical compromise implies a logical compromise.

**Compromised node geolocation**

If a location-aware node has been compromised, its location information becomes available to the attacker. If it is not location-aware, once a node has been compromised, an attacker can force it to generate an innocuous transmission, and rely on the techniques for geolocating a transmitting node in order to geographically locate that node.

If the surveillance equipment required for transmitter geolocation is not available, or if anti-surveillance protection (i.e., TRANSEC) measures have been taken, an attacker can periodically and subtly vary a compromised node’s display [61], or cause it to emit inaudible chirps [64], thereby covertly turning it into a detectable [86][95] beacon.

**Wormholing**

An attacker can transfer traffic observed in one area of a MANET into another, possibly using a low latency backbone network [46]. This can disrupt the workings of any MANET protocol, potentially including those designed to provide anonymity.

**2.4.2 Modi operandi**

We list several ways in which the above capabilities can be used and combined.

**Node implication**

An attacker-owned server can learn the network layer addresses of its clients. If the attacker can individually inspect all MANET nodes before they remove traces of using a certain network layer address, he is able to learn which of the nodes accessed the server. This is useful to an attacker in a situation where some questionable content is stored on a steganographic filesystem on the client and cannot be proved to exist by any other means. The same applies if the attacker owns the client and is trying to implicate the server.
Location service utilization

A location service allows one to geographically locate nodes in the network, according to services that they offer or according to an identity of theirs. An attacker can utilize such a location service to locate and possibly track a target node and its motion over time.

As mentioned, on the Internet, ISPs offer location services to law enforcement agencies wishing to locate a user at a given IP address. In cellular networks, operators offer location services for the sake of locating the origin of an emergency call and for the sake of offering users location-based applications [13]. In MANETs with location-aware devices, a location service can be built in-band, to support search and rescue operations, for example.

Iterative node geolocation

In addition to utilizing a location service, an attacker can learn and arrive at the location of a target node by learning a current route to that node (e.g., via traceroute). Once route information is available, the attacker can use the information in conjunction with node compromise and transmitter geolocation methods to iteratively compromise and geolocate all the nodes in the route, in real time, until arriving at the location of the target node.

Besiegement

An attacker can geographically besiege nodes in an area. Given eavesdropping capabilities, the attacker learns the information going in and out of the besieged area. Given man-in-the-middle capabilities, the attacker controls the information going in and out.

An attacker can logically besiege a portion of the network by compromising all of its surrounding neighbors. If the attacker knows that the besieged nodes have no other avenue of communication, he can detect for certain when they transmit information to or receive information from the outside world. If the attacker knows that a besieged area contains only one device, he is able to attribute communications in that area to it.

Traffic analysis

Once an eavesdropper has the network layer information of who communicates with whom, he can use it to reveal deeper relationships between communicating nodes and to learn of the function or role of given nodes in the network. One example is to use the information to analyze traffic volumes. A node with a high volume of traffic is likely to be a node of importance and a potential target for a denial of service attack or for logical compromise. An attacker can also use the information to better place himself, either geographically or in terms of node compromise, in order to besiege a certain portion of the network.

Motion analysis

An attacker can detect motion by setting up eavesdropping equipment and waiting for the same network layer address to be used in multiple locations. Assuming that rudimentary
geolocation of the transmissions is possible, the attacker can track the motion of a node that uses a static address.

If many devices move in the same pattern in relation to the attacker, this type of motion analysis does not suffice to resolve them. For instance, with less than perfect spatial eavesdropping resolution, an attacker can only observe many static addresses reappearing in many observed locations, but which address belongs to which device is unclear unless the attacker can cross-reference the data with auxiliary information. In particular, with zero spatial eavesdropping resolution, motion detection of the above type is impossible.

**Advanced motion analysis**

When a node’s motion pattern in relation to an attacker *is* unique with respect to the attacker’s spatial eavesdropping resolution, an attacker can use an intersection attack, in which the repeatedly appearing logical address is correlated to a node (or a few nodes) that were observed to be in all of the locations in which it (the address) was observed. Somewhat less trivially, an attacker may install a sensor network and rely on collected packet counts to detect node locations and motion [99], without the need to observe logical addresses.

**Route distortion**

In an insecure routing protocol, a local attacker can exploit weaknesses in the protocol in order to cause constructed routes to include him as an intermediate node [7][19]. In this manner, the attacker can learn the contents of and modify packets that would not normally traverse his area of the network. In particular, if route members are assumed to be trusted or if identifying information is passed in the clear, then the attacker succeeds.

**Timing and flow analysis**

If an eavesdropper can link logical addresses to devices, then he can observe the time intervals during which data flows from one device to another, and conclude that two devices with matching time intervals are communicating with each other (timing analysis [65][72][96]).

A more active attacker can intentionally delay messages to make relationships between communicating nodes more detectable (flow marking [33]), or inject data into the network, in an effort to detect which nodes are more busy communicating than others or to approximate the distance between nodes by measuring relative response delays.

Alternatively, an attacker can observe characteristics such as message size (akin to packet counting [6]) or utilized compression algorithm, in order to link a sender to a receiver.
2.5 Adversary model

We model an adversary, $A$, in terms of his view of system executions. Given an execution, $\alpha$, we define $\text{view}^A_\alpha$ as a function over $\text{emissions}_\alpha$. Together with a-priori information about the observed execution, the definition of $\text{view}^A_\alpha$ establishes the strength of an attacker\(^3\).

For instance, if $\text{view}^A_\alpha$’s image contains the execution’s delivery events, we are modeling an attacker that knows the sender, recipient and value of every message, like the one considered by Chaum in [15]. Such an attacker knows what data is sent or received by which device in the MANET, and corresponds to a global eavesdropper with perfect fingerprinting resolution and the ability to link messages to devices (e.g., keeps track of device locations at all times and has perfect spatial eavesdropping resolution). If $\text{view}^A_\alpha$’s image further contains timestamps, we are modeling an attacker that can time observed events.

Or, for instance, if $\text{view}^A_\alpha$ maps emissions made at different locations to images with fewer (or no) associated locations, then $\text{view}^A_\alpha$ “loses information” and we are modeling an attacker with limited spatial eavesdropping resolution. Note that $\text{view}^A_\alpha$’s domain contains elements of both $\text{COI}$ and $\text{UOI}$. This captures the notion that an attacker may use all observable items (not just algorithmically controlled ones) to break system anonymity.

If $\text{view}^A_\alpha$’s image contains $\text{emissions}_\alpha$, then the attacker is a global attacker. Otherwise, the attacker is a local attacker. If the attacker cannot influence the contents of $\text{emissions}_\alpha$, then he is a passive attacker. Otherwise, he is an active attacker. If the attacker owns a node in the system, he is an internal attacker. Otherwise, he is an external attacker.

An attacker’s view induces an observational equivalence relation on observable items, such that observable items are deemed equivalent if they “appear the same” to the attacker in every execution. More formally, the relation is denoted $\equiv_A$ and is defined as follows:

$$\equiv_A = \{(oi_1, oi_2) \in OI^2 \mid \forall \alpha \quad \text{view}^A_\alpha(oi_1) = \text{view}^A_\alpha(oi_2)\}$$

The resulting partition of the set of observables can intuitively be thought of as a set of “colors” discernible by $A$. Correspondingly, we dub it $C_A = OI/\equiv_A$. In the next chapter, we set aside views and define anonymity and related notions directly in terms of $C_A$.

Ultimately, an attacker’s goal is to learn or otherwise manufacture a node-dependent bias, which would allow him to resolve nodes and attribute observed actions to each of them individually. Otherwise, if an observed action warrants retaliation, the attacker is forced to retaliate against all nodes in the system, which may be prohibitively expensive. We note that resolution is necessary but not sufficient for attribution of actions to individuals.

In analyzing the algorithms presented in this work, we assume a passive, external, global and computationally unbounded adversary, for which $\text{view}^A_\alpha$ contains the addresses used by nodes, but not the execution’s delivery events. For instance, one may envision an adversary in the business of signals intelligence (i.e., SIGINT). The adversary may control a reconnaissance satellite whose coverage permits packet inspection, but whose ability to accurately pinpoint transmissions is limited. For the sake of simplicity, we assume that the adversary has zero spatial eavesdropping capability. In addition, much more reluctantly, we ignore time and assume that the adversary cannot sequence eavesdropped messages.

\(^3\)A view-based definition of attacker strength such as this one agrees with the common definition of an adversary as a function in the context of distributed systems [5][67]. Given such a definition, one can represent an adversary’s partial knowledge of the system using function views and their opaqueness [47].
2.6 Trust model

Trust is anchored in shared information, which is placed in nodes ahead of time or is dynamically acquired during an execution. System designers must ensure or assume that trust is well-placed and is established without impeding anonymity at any stage.

In the context of point to point communication systems, examples of trust include the use of pre-assigned and unique network layer addresses [82][45] and the use of trusted name resolution servers [69]. Trust is also generally used for bootstrapping security [83][3].

In the context of anonymous communications on the Internet, examples of trust include the use of pre-shared secrets (e.g., account passwords in Crowds [84]) and the use of globally known, securely distributed, public keys (e.g., mixes [15], their descendants [100][32] and other PKI-based schemes [37][24]). Less common are examples of dynamically acquired trust, as in the use of secret sharing techniques to avoid unwieldy key distribution [54].

In our main contributions, presented in chapters 6 and 7, we assume that every node trusts every other node separately from every other node. That is, nodes anonymously establish pairwise trust relationships, either in advance or dynamically during execution.

We consider this to be the least restrictive trust model suitable for the purpose of building a distributed point to point communication system. On the one hand, the model does not assume the existence of widely known information (“widely” could mean globally or even locally for a given neighborhood), yet still allows all nodes to trust one another enough to individually communicate with one another. On the other hand, the model does not assume the existence of a single trusted authority, which would require extra security measures to protect, may be difficult to scale and robustly maintain, does not offer a natural fit for MANETs that form independently of any central authority, nor fit the bill for participants unwilling to place their anonymity in the hands of a third party.\(^4\)

2.7 Motion model

Although node motion is a characteristic feature of MANETs, and despite its potential effect on anonymity, we do not specify a motion model in this work. As in our adversarial model of section 2.5, let us ignore spacetime, for the sake of simplicity of exposition.

\(^4\)Which, as an egg-gathering basket, may be subject to subpoena or other legal (or illegal) actions.
Chapter 3

Goals

In this chapter, we present our three primary goals of system design, in order of decreasing significance. In the process, we introduce definitions that permit system specification and analysis in subsequent chapters. We conclude by stating the problem to be solved.

The first and foremost goal is to design a system that is “anonymous”. To this end, we give a formal definition of the “anonymity” of a system and a method for comparing the “anonymity” of different systems. The second goal is to design a functioning point to point communication system. To this end, we introduce a template for routing algorithm procedures and describe their use of identities. The third goal is to design a system that is efficient. To this end, we state desirable performance properties in an informal manner.

3.1 Anonymity (and friends)

In order to design an “anonymous” system, we first characterize system “anonymity” and related concepts. There exist several related notions of information hiding that bear on the ability to identify processors in a distributed system, and the “anonymity” that is achieved by a given scheme can be defined and measured in more than one way [84][44][78]. Where unambiguous, we use the informal term “anonymity” (as in “system anonymity” and “unconditional anonymity”) as an umbrella term encompassing these notions.

As a form of information hiding, anonymity is studied within the framework of probability theory. In this work, we give boolean information-theoretic definitions of anonymity akin to those of unconditional security. Thus, an unconditionally anonymous system can withstand a computationally unbounded adversary. More fine-grained information-theoretic metrics of anonymity are given and applied to existing systems in [90] and [22].

Consider an adversary, $\mathcal{A}$, the observational equivalence relation induced by his view, $\equiv_{\mathcal{A}}$, and the resulting partition of the set of observables, $\mathcal{C}_{\mathcal{A}}$. Given an algorithm, $\text{ALG}$, an input to it, $x \in X$, and an equivalence class, $c \in \mathcal{C}_{\mathcal{A}}$, we define $E_{x \rightarrow c}^{\text{ALG}}$ to be the event where execution of $\text{ALG}$ on $x$ results in emissions observed as the “color” $c$. Thus, an algorithm induces an indistinguishability relation on its inputs, $\sim$, which is defined as follows:

$$\sim \doteq \{(x_1, x_2) \in X^2 \mid \forall c \in \mathcal{C}_{\mathcal{A}} \quad P_{\text{r}[E_{x_1 \rightarrow c}^{\text{ALG}}]} = P_{\text{r}[E_{x_2 \rightarrow c}^{\text{ALG}}]}\}$$

In the following subsections, definitions of anonymity are stated in terms of $\sim$. The section concludes with a definition of an information hiding order on distributed systems.
3.1.1 Unobservability

Unobservability is defined as the analog of perfect secrecy – system emissions must be independent of inputs and state, such that observation of the former teaches the attacker nothing about the latter\(^1\). More tenably, all inputs are required to be indistinguishable.

**Definition 1** An algorithm achieves **unobservability** if

\[
\forall x_1 \forall x_2 \left[ x_1 \sim x_2 \right]
\]

We note that unobservability is “easier” than secrecy, because it does not impose the constraint of unique decryption. Hence, a trivial example of achieving unobservability is that of a system whose emissions are constant regardless of its input and internal state.

3.1.2 Immeasurability

Instead of hiding everything about the input, one can settle for hiding particular attributes only, such as the number of active nodes or the number of sent messages. In terms of the analogy to secrecy, the attribute is a part of the “plaintext” and must not be leaked.

An attribute is modeled by a function, \( f : X \rightarrow V \), which gives an attribute value per input. Immeasurability requires emissions to reveal nothing of the value possessed by the input. More tenably, every value must be represented in every indistinguishability class.

**Definition 2** An algorithm achieves **\( f \)-immeasurability** if

\[
\forall v \exists x_1 \exists x_2 \left[ x_1 \sim x_2 \land f(x_2) = v \right]
\]

Immeasurability is a special case of unobservability and is closely related to the German Tank Problem [39], in which an adversary tries to estimate the number of manufactured tanks given the observed serial numbers of a subset of tanks. We note that a well-studied field applicable to this problem is that of animal population size estimation (e.g., mark and recapture approaches [88][89], including Lincoln-Peterson [66][77] analysis). We also note that with zero fingerprinting resolution (e.g., tanks do not have visible serial numbers, or tanks get relabeled when the adversary is not watching), enumeration becomes harder.

Theoretically, any algorithm can be turned into one that achieves immeasurability with respect to node count, given that a single processor can be made to simulate any number of others. Given a system with a certain node count, one can construct systems of arbitrary node counts that behave in an identical manner, with few nodes simulating many or vice versa. Unfortunately, practically speaking, physical constraints may prevent a high fidelity simulation (e.g., given a sufficiently powerful adversary, a processor cannot appear to be in multiple locations simultaneously, nor can a slow processor simulate a fast one).

---

\(^1\)Notably, our definition contains existing definitions of unobservability (see [79] and [44]) as a special case, since the existence of real (as opposed to dummy) messages is part of the processors’ hidden states.
3.1.3 Anonymity

We begin by elucidating potentially conflicting terminology. In the context of distributed systems, an *anonymous* system or network is one in which all processors share the same state machine [5]. In the context of anonymous communication, it is implicitly assumed that the underlying distributed system is not anonymous, or else there would be no need for an anonymization scheme. In this work, where potentially confusing, we refer to the anonymity achieved by an anonymization scheme as *observational* anonymity.

Disregarding differences such as form factor, hardware and software configuration, or location, all nodes are essentially “isomorphic” to one another at their state machine core, with nothing truly identifying one from another; only their unique characteristics allow for their identification. Given “isomorphic” nodes, an attacker cannot detect when one node is being clandestinely swapped for another, nor in turn attribute an observed action to any but all of them. Conversely put, if node permutation has no observable effect, then nodes cannot be meaningfully named by said attacker, thereby securing their anonymity.

Such an ideal motivates us to define observational anonymity as the indistinguishability of isomorphic inputs. We denote isomorphism under node permutation with \(\simeq\), such that \(x_1 \simeq x_2\) if and only if \(x_2\) can be obtained from \(x_1\) by applying a permutation of \(P\) to it. Equivalently, \(x_1\) and \(x_2\) are such that the roles played by particular nodes are reversed.

**Definition 3** An algorithm achieves *observational anonymity* if

\[
\forall x_1 \forall x_2 \left[ x_1 \simeq x_2 \Rightarrow x_1 \sim x_2 \right]
\]

One can think of observational anonymity as a measure of the observed symmetry [52] of a system, or the observed similarity of its processors. In this light, the purpose of an anonymization scheme is not necessarily to hide nodes’ logical identities, but rather to “smooth out” underlying dissimilarities between processors by superficially presenting asymmetric unobservables (such as communication patterns) as though symmetric. Thus, our goal is to ensure that system emissions remain invariant under node permutation.

The flexible and contextual nature of isomorphism lets us model various relaxations of observational anonymity with equal ease. By substituting various isomorphism relations, one can relax the requirement above to specifically refer to senders, receivers, leaders, route members, or nodes harboring questionable content. In particular, we informally define

1. **Sender anonymity**: inputs with differing senders are indistinguishable.
2. **Receiver anonymity**: inputs with differing receivers are indistinguishable.
3. **Relationship anonymity**: inputs with differing relationships are indistinguishable. An attacker cannot learn which nodes in the system are indicating one another as destinations and accepting messages from one another, despite perhaps perfectly resolving all communicating devices and eavesdropping on their messages at all times.
4. **Location anonymity**: inputs with differing locations are indistinguishable. An attacker cannot distinguish nodes based on location information (for instance, when all the nodes have the same probability of appearing at any location at any time). As shown in section 2.4, location anonymity can be compromised by lack of location privacy.
3.1.4 Comparison

The definitions given above fall on a scale of information hiding. Intuitively, the greater the number of indistinguishable inputs, the larger the size of the equivalence classes induced by \( \sim \), the smaller their overall number and the less information an attacker has.

In the case of unobservability, every input is in a class with all other inputs. In the case of immeasurability, every input is in a class with inputs having all possible attribute values. In the case of anonymity, every input is in a class with all other inputs isomorphic to it.

Since different applications have different information hiding requirements, a universal measure of information hiding is not necessarily applicable (what is good enough for one is not necessarily good enough for another). Instead, let us opt to compare algorithms by comparing the partitions that they induce on the input space. If the partition induced by \( ALG_1 \) refines that of \( ALG_2 \), we write \( ALG_1 \preceq ALG_2 \). If the refinement is proper (i.e., there exists an input pair in \( \sim_{ALG_2} \) that is not in \( \sim_{ALG_1} \)), we write \( ALG_1 < ALG_2 \).

3.2 Communication (and routing)

In a distributed system, it is often desirable for nodes to copy information from one to another. When information is copied from \( p_i \) to \( p_j \), we say that \( p_i \) communicates with (or sends a message to) \( p_j \). Our goal is to facilitate pairwise copying of information among nodes, by designing a point to point communication system for that very purpose.

Conceptually, the input to a point to point communication system is a directed multigraph with an arc per message to be sent. For the sake of simplicity, we ignore message contents and the order in which they are sent. Hence, an input may look as follows:

![Diagram of a point to point communication system](image)

A communication system attempts to ensure that information is copied as specified by the input (i.e., that messages are delivered at their destinations). In the example above, \( p_1 \) must successfully send one message to \( p_2 \) and two messages to \( p_3 \). Similarly, \( p_2 \) must successfully send a message to \( p_3 \), who sends a message to itself. Given the uncertainty of the network, delivery is far from assured. Still, we pay little mind to reliability guarantees, as they appear orthogonal to anonymity and fall outside of network layer jurisdiction.

Though we model inputs as multigraphs, multigraphs and names à la “\( p_1 \)” or “\( p_2 \)” are never directly represented in the system. Given the distributed nature of the system, input must formally be specifiable as a sequence of per-node input events. Consequently, an input multigraph is first translated into per-node routing algorithm procedure invocations.

In the following subsections, we present a template for routing algorithm procedures (which is further used for algorithm specification in subsequent chapters) and characterize the relationship between an input multigraph and invocations of these procedures. Then, we proceed to describe identities and their use in identity-based routing algorithms. For the interested reader, optional routing algorithm features are discussed in Appendix A.
3.2.1 Procedures

A network layer routing algorithm defines two procedures per node, \texttt{Network-Layer-Send} and \texttt{Network-Layer-Recv}. \texttt{Network-Layer-Send} is invoked by the transport layer of a node when it wishes to originate a message to some node in the network. Its arguments are the newly originated message and an indication of the intended recipient. \texttt{Network-Layer-Recv} is invoked by the datalink layer of a node when it is done processing a frame that has been received on some link. Its argument is the received message, stripped of lower layer data.

A template for these two procedures is given in Figures 3.1 and 3.2, respectively. A routing algorithm built according to this template must define the procedures

\begin{description}
  \item[Route-On-Send:] routes a newly originated message to its destination.
  \item[Filter:] indicates whether an incoming message should be ignored by the processor.
  \item[For-Me:] indicates whether an incoming message is intended for the processor.
  \item[Accept:] handles an incoming message intended for the processor, delivers data.
  \item[Route-On-Recv:] routes an incoming message intended for another.
\end{description}

3.2.2 Invocations

In this subsection, we describe how an input multigraph is translated into invocations of \texttt{Network-Layer-Send}. In the subsection that follows, we describe identities and their use.
A sender must specify a destination in order to send a message to it. In our template, this is indicated by the second parameter to Network-Layer-Send. A target value serves as a private representation of a destination, meaningless outside of node-local context. We note that a sender must store as many target values as there are recipients it wishes to simultaneously address – on account of the pigeonhole principle, \( k \) recipients cannot be uniquely addressed with less than \( k \) state. In a network of \( n \) nodes, this implies a space requirement of \( O(n^2 \log n) \) in the worst case (all nodes wish to unicast to all others simultaneously), regardless of the routing algorithm and representation being used.

The mechanism by which destinations are specified (i.e., how parameters to Network-Layer-Send are derived) is an integral part of a routing algorithm, as is detailed below. In contrast, the mechanism by which destinations are chosen (i.e., how an input multigraph is derived) is outside the scope of this work. We assume that node choices and communication patterns are not directly observable, or else anonymity is a lost cause to begin with.

Let \( G \) be an input multigraph. Let \( T_1 \) be the set of target values at \( p_1 \). Let \( T = \bigcup T_i \). A routing algorithm translates \( G \) into invocations of Network-Layer-Send by computing an arc labeling, \( \tau: E(G) \rightarrow T \), in which arcs leaving \( p_i \) are labeled with elements of \( T_i \). The labeling matches messages to be sent with parameter values to be used during invocation. Without loss of generality, let \( T_i = \{ m_{ei}, 1, 2, \ldots \} \), where \( m_{ei} \) is used by a node to represent itself. Considering the example input from before, the labeling \( \tau \) may look as follows:

Metaphorically, a routing algorithm provides senders with buttons to press, such that pressing a particular button (i.e., invoking Network-Layer-Send with a particular target value) ideally results in message delivery at a particular destination. Given our example \( \tau \) above, \( p_1 \) would press \( 1_1 \) in order to reach \( p_2 \) and either \( 2_1 \) or \( 3_1 \) in order to reach \( p_3 \). Similarly, \( p_2 \) would press \( 1_2 \) in order to reach \( p_3 \), who would press \( m_{e3} \) in order to reach itself.

As noted, the main purpose of a routing algorithm is to establish pairwise channels, such that any message can be sent by any node to any other. Throughout the rest of this work, we limit our focus to one particularly popular channel setup mechanism – that of identities. In addition, we limit our focus to symmetric algorithms, in which all nodes use the same (parametric in identities) implementations of routing algorithm procedures.

### 3.2.3 Identities

In order to convey a choice of destination across the network, routing algorithms customarily derive a destination indicator and include it with the sent message. A destination indicator is a public representation of a destination, which is sent to others over the network\(^2\).

In traditional routing algorithms, the destination indicator takes the form of a static, fixed-length value, also known as a network layer address. However, this is not a strict

\(^2\)At times [75][74][51][40], the distinction between the target value and the destination indicator is blurred, and the same value is taken for both and used directly in routing algorithm data structures.
requirement, and in fact, may be suboptimal for MANETs [11][29]. In position-based routing algorithms [36], the destination indicator takes the form of location information.

When routing the message, the destination indicator is used in order to break symmetry, such that only the intended recipient performs a certain action (e.g., delivers the message to the upper layers). In our template above, this is indicated by the branching structure of Network-Layer-Recv, which accepts a message only when For-Me decrees that it should.

For such a scheme to work, nodes must coordinate the use of destination indicators. In this work, we assume the global use of a single set of destination indicators, \( A = \{1, 2, \ldots\} \).

Coordination is achieved by computation of an implicit arc labeling, \( \kappa: E(G) \rightarrow A \), which matches messages to be sent with agreed-upon destination indicators to be used in their transmission. Given the example input from before, the labeling \( \kappa \) may look as follows:

\[
\begin{array}{c}
\text{p}_1 \xrightarrow{15} \text{p}_3 \\
\text{p}_2 \xrightarrow{3} \text{N/A} \\
\text{p}_3 \xrightarrow{92} \end{array}
\]

As our choice of symbols suggests, \( \kappa \) gives the addresses, aliases, or appellations that nodes are known by. Given our example \( \kappa \) above, \( \text{p}_1 \) knows \( \text{p}_2 \) by the alias 3. Subsequently, it includes this alias with its message to \( \text{p}_2 \). Much the same, \( \text{p}_2 \) knows \( \text{p}_3 \) by the alias 14. In addition, \( \text{p}_3 \) is also known by the aliases 15 and 92, which are separately included with the messages from \( \text{p}_1 \). Without loss of generality, we assume that a node does not need an alias to contact itself and, here and henceforth, disregard arcs from a node to itself.

In order to enable communication, the agreement \( \kappa \) must be embedded in the nodes. For this purpose, an identity-based routing algorithm defines an extra pair of functions at every \( p_i \), which map private destination representations to public ones and vice versa:

\[
\text{Encode-Destination-Indicator}_i: T_i \rightarrow A \\
\text{Decode-Destination-Indicator}_i: A \rightarrow T_i
\]

An identity-based routing algorithm utilizes the two functions to ensure that messages to \( \text{p}_j \) are sent with addresses of \( \text{p}_j \), such that \( \text{p}_j \) accepts them upon reception. On the one end, \text{Encode-Destination-Indicator} lets a sender compute an address to be included with a message (i.e., when executing Route-On-Send). On the other, \text{Decode-Destination-Indicator} lets others decide whether a message is intended for them (i.e., when executing For-Me).

The two functions are derived from \( \kappa \) as follows. Observe an arc \( e \) from \( p_i \) to \( p_j \). First, we set \( \text{Encode-Destination-Indicator}_i(\tau(e)) = \kappa(e) \), such that \( p_i \) knows \( p_j \) by the alias \( \kappa(e) \). Then, correspondingly, we set \( \kappa(e) \) to be an identity of \( p_j \). We conclude this section with a demonstration of the above. By our convention, an address, \( a \in A \), is an identity of \( p_j \) if \( \text{Decode-Destination-Indicator}_j(a) = \text{me}_j \). Given our previously depicted \( \tau \) and \( \kappa \), the two functions are defined as follows, in accordance with the four arcs of the multigraph:

\[
\text{Encode-Destination-Indicator}_1(1_1) = 3 \quad \text{and} \quad \text{Decode-Destination-Indicator}_2(3) = \text{me}_2 \\
\text{Encode-Destination-Indicator}_1(2_1) = 15 \quad \text{and} \quad \text{Decode-Destination-Indicator}_3(15) = \text{me}_3 \\
\text{Encode-Destination-Indicator}_1(3_1) = 92 \quad \text{and} \quad \text{Decode-Destination-Indicator}_3(92) = \text{me}_3 \\
\text{Encode-Destination-Indicator}_2(1_2) = 14 \quad \text{and} \quad \text{Decode-Destination-Indicator}_3(14) = \text{me}_3
\]
3.3 Performance

In this section, we list desirable performance goals. While not all goals are accounted for by the algorithms in this work, we list them here for the benefit of future designs.

3.3.1 Complexity

A system ought to achieve minimal (asymptotic) space and communication complexity. Space complexity is measured in the number of bits stored per node. In particular, we wish to minimize the number of addresses and the size of routing tables stored at every node. Communication complexity is measured in the number of datalink messages sent. In particular, we wish to minimize the number of messages flooded in the network.

3.3.2 Cryptography

A system ought to achieve maximal compatibility between cryptographic primitives and the scenario at hand. Unless required, secrecy and integrity are not provided. Or, if nodes are computationally weak, lightweight cryptography is preferable. In MANETs especially, we wish to minimize trust in third parties, the number of random bits to be generated and the required cryptographic operations per message (computational complexity overhead).

3.3.3 Heterogeneity

A system ought to achieve maximal consideration and utilization of node heterogeneity. Heterogeneity includes uneven battery power, uneven memory utilization and uneven processor speeds, among others. When possible, work is distributed fairly among nodes.

3.3.4 Implementability

A system ought to be easy to implement and testable in a simulator.

3.3.5 Robustness

A system ought to achieve maximal fault tolerance and adaptability to dynamic conditions. In particular, we wish to reduce vulnerability to denial of service, tolerate failure of crucial nodes, perform well under node churn and minimize the cost incurred by node motion.

3.3.6 Scalability

A system ought to achieve good performance at large scales. In particular, we wish to prevent the formation of bottlenecks, to cope with large scale motion patterns and to distribute knowledge in the network promptly and to the extent required.
3.4 Problem statement

Given a network scenario and an adversary of a given strength, design a functioning point to point communication system that is observationally anonymous and efficient. The system must achieve selected anonymity properties in a manner sufficient to foil the adversary.

At the one extreme, if the adversary is blind, then any design will do. At the other extreme, if the adversary is omniscient and capable of revealing the input, then no design will do. In between these extremes, one must design a system that exhibits symmetry with respect to the adversary’s view, ensuring as many indistinguishable inputs as possible. Although one is free to choose the design, one is not allowed to make assumptions about the likelihood of different inputs, as the likelihood of inputs is prescribed on a per-application basis.
Chapter 4

Anonymity-blasé systems

Let us begin our investigation with two existing and well-known identity-based routing algorithms that were designed without anonymity in mind. Our motivation in examining anonymity-blasé systems is to learn about the anonymity that is inherent in a MANET, before applying additional anonymization techniques. In this chapter, we specify the two algorithms according to the template and terminology presented in the previous chapter. We also list their assumptions about the network, the adversary and trust in the system. In chapters 5, 6 and 7, we analyze their anonymity and present our contributions.

Due to their similarities, we specify identity-based routing algorithms parametrically. We assume the existence of a distributed agreement algorithm, GenID, that computes $\kappa$. Implementation of GenID, be it in or out of band, is outside the scope of this work. An algorithm is specified in terms of $\kappa$, such that any agreement algorithm can be “plugged in” without a change in specification. Put differently, the question of arriving at an agreement is orthogonal to that of using that agreement for routing. In future chapters, we study the anonymity of the two algorithms of this chapter under various agreement algorithms. An algorithm, ALG, that uses an agreement algorithm, GenID, is denoted ALG(GenID).

Once an agreement has been forged, it is embedded in the nodes, as demonstrated in the previous chapter. At the one end, Decode-Destination-Indicator is given by an allocation protocol, whose outcome is a set of identities of $p_i$, at every $p_i$. Without loss of generality, we assume that identities are allocated in advance or upon joining the network. Allocation can be conducted by a central authority or in a distributed and autonomous manner. We also assume that, with high probability, no address is allocated to more than one node. At the other end, Encode-Destination-Indicator is given by a dissemination protocol, whose outcome is a set of identities of recipients, at every sender. The mechanism by which identities are disseminated, be it by hand or by naming server, is outside the scope of this work.

4.1 Flood Routing Algorithms

A flood routing algorithm, or FRA, is a most simple algorithm. An FRA performs a series of datalink layer broadcasts in every neighborhood, originating at the sender and propagating throughout the network. A node that receives a never-before-seen message delivers it to the upper layers if it is the intended destination, or otherwise continues to broadcast it.
4.1.1 Assumptions

Network

- Upon joining a network, a node is allocated a unique address from a global pool. The address remains unique for as long as the device remains in the network.
- The address allocated to a recipient is used by senders as a destination indicator. Optionally, a sender can include its own address when a two-way session is desired.
- All nodes serve both as endpoints and as routers.

Adversary None. The system was designed without an adversary in mind.

Trust

- A node must know the address of its desired partner in order to send a message.
- Any node can send messages to and receive messages from any other node.

4.1.2 Specification

Implementations of Route-On-Send, Filter, For-Me, Accept and Route-On-Recv for FRAs are shown in Figures 4.1 through 4.5, respectively. They are explained below:

1. FRA-Route-On-Send broadcasts the newly originated message to all neighbors. The message is sent along with a destination indicator and a flood indicator. Trivially, the destination indicator indicates the destination. The flood indicator allows others to determine whether they have already flooded the message. One might imagine many ways of generating flood indicators. In this work, implementation of flood indicator generation is delegated to Encode-Flood-Indicator and remains outside the scope.

2. FRA-Filter discards previously received messages, to prevent excessive flooding.

3. FRA-For-Me checks whether the destination indicator listed in the message is one that has been assigned to the local node.

4. FRA-Accept delivers the message to the upper layers.

5. FRA-Route-On-Recv broadcasts a message to all neighbors.

The implementations of FRA-Route-On-Send, FRA-Filter and FRA-Route-On-Recv can also be extended to limit the spread of flooding to a certain number of hops. Note, however, that naive use of a hop counter may divulge the distance in hops between nodes, thereby leaking information about the datalink layer (if not physical) topology graph.

Destination representations are handled in FRA-Encode-Destination-Indicator and FRA-Decode-Destination-Indicator, shown in Figures 4.6 and 4.7, respectively. Together, these two procedures encapsulate the handling of network layer identities in the algorithm.
Procedure FRA-Route-On-Send($m$: message, $t$: target)
begin
    $M := \text{new message}$
    $M.msg := m$
    $M.fid := \text{Encode-Flood-Indicator}(m, t)$
    $M.dest := \text{Encode-Destination-Indicator}(t)$
    call Datalink-Broadcast($M$)
end

Figure 4.1: Route-On-Send() Procedure for FRAs

Procedure FRA-Filter($m$: message)
 Initially, $\text{Seen} = \emptyset$
begin
    if $m.fid \in \text{Seen}$ then
        return $true$
    $\text{Seen} := \text{Seen} \cup m.fid$
    return $false$
end

Figure 4.2: Filter() Procedure for FRAs

Procedure FRA-For-Me($m$: message)
begin
    $t := \text{Decode-Destination-Indicator}(m.dest)$
    if $t \neq me$ then
        return $false$
    return $true$
end

Figure 4.3: For-Me() Procedure for FRAs

Procedure FRA-Accept($m$: message)
begin
    deliver $m.msg$
end

Figure 4.4: Accept() Procedure for FRAs
Procedure FRA-Route-On-Recv\((m: \text{message})\)
begin
  call Datalink-Broadcast\((m)\)
end

Figure 4.5: Route-On-Recv() Procedure for FRAs

Procedure FRA-Encode-Destination-Indicator\((t: \text{target})\)
Let \(\tau: E(G) \rightarrow T\) be the private arc labeling of \(G\)
Let \(e \in E(G)\) be an arc from \(p_i\), for which \(\tau(e) = t\)
Let \(\kappa: E(G) \rightarrow A\) be the public arc labeling of \(G\)
begin
  return \(\kappa(e)\) /* receiver address, obtained anonymously somehow */
end

Figure 4.6: Encode-Destination-Indicator() Procedure for FRAs

Procedure FRA-Decode-Destination-Indicator\((a: \text{address})\)
Let \(\kappa: E(G) \rightarrow A\) be the public arc labeling of \(G\)
Let \(IDs_i = \kappa(in_i)\), where \(in_i\) is the set of arcs to \(p_i\)
begin
  if \(a \in IDs_i\) then /* optional use of Bloom filter [9] to save space */
    return \(me_i\)
  else
    return \(\bot\) /* private representation of \(a\) is unavailable */
end

Figure 4.7: Decode-Destination-Indicator() Procedure for FRAs
4.2 Dynamic Source Routing (DSR)

In this work, we describe a skeleton version of the Dynamic Source Routing Protocol [51], without optimizations such as route reversal at a receiver (designed for easing return traffic) and route caching (designed for utilizing other nodes’ knowledge of routes), and without route maintenance features such as route error messages and local repair. Since the phases of DSR can each be modeled as a variation of flood routing, throughout the rest of this work, we conjecture that results attained for FRAs apply equally well to DSR.

4.2.1 Assumptions

Network, adversary and trust assumptions are the same as in FRAs.

4.2.2 Specification

We describe DSR as a three phase protocol. For the sake of reference, implementations of Route-On-Send, Filter, For-Me, Accept and Route-On-Recv are listed in Appendix B.

In the first phase, the sender floods a route request to reach its desired partner. Routers add their own addresses to the route request and forward it. Assuming no malicious behavior, the real destination node receives the request, with accrued path information.

In the second phase, the receiver issues a route reply. For the sake of simplicity, we assume that the receiver replies with the obtained path information, unmodified. The reply is routed back to the sender, who stores the hop sequence contained in it, for later use.

In the third phase, the sender uses the stored sequence as the source route field of any data message sent to the destination, allowing the message to be forwarded accordingly.

In DSR, the localization of knowledge concerning routers is an extreme one – an address that is added to a route request must later be recognized in the route reply and data transfer phases, but only by its owner. Since routers are only required to recognize their own router indicators, router indicators can be chosen locally per node, without node coordination.
Chapter 5

Analysis

Since the performance properties of the above two algorithms are well-known, we analyze their information hiding properties only. In section 5.1, we present our initial observations concerning anonymity in MANETs, upon which our analysis is based. In section 5.2, we discuss the limitations of our analysis. In section 5.3, we give required preliminary definitions. The analysis itself is then presented in two parts, in chapters 6 and 7, respectively.

5.1 Observations

Recall $D$ as the set of all devices. As system designers, we ourselves can always distinguish a device in a MANET according to a (universal) device identity. Thus, we define the identity of a device to be its (universal) device identity as seen from our perspective, not to be confused with a datalink layer identity, a network layer identity, a hardware serial number or a device’s identity as surmised by an attacker. More formally, we choose an index set of $D$ and define a (universal) device identity to be an element of that set. With device identities defined, let us make the following observations concerning anonymity:

1. The attacker considered by Chaum in [15] corresponds to a global eavesdropper with perfect fingerprinting resolution and the ability to link messages to device identities (for instance, by possessing perfect spatial eavesdropping resolution and the ability to track device locations). Such an attacker knows what data is sent or received by which device. We do not offer protection against this brand of omniscience.

2. Importantly, device and user identification is not required when joining a MANET. No less importantly, a network layer identity can consist of arbitrary data and is not necessarily tied to a device. Therefore, a network layer identity does not imply device identity in and of itself. In addition, unless specifically dealing with position-based routing schemes, a network layer identity does not imply device location, as it possibly would on the Internet. Hence, there is no need to hide a network layer identity.

3. A logical route is a sequence of network layer identities. As a corollary of the above observation, a logical route does not imply device identities of routing devices in and of itself. We distinguish between a logical route and a physical route, which is a sequence of devices (universal device identities) through which a given message is routed.
4. Based on the above, we conclude that hiding the contents of logical routes from an attacker does not significantly increase his difficulty in resolving devices, although it may increase his difficulty in logically accessing (and compromising) them.

5. An attacker with zero spatial eavesdropping resolution makes an inference of the type “nodes $p_1$ and $p_2$ both communicated with node $p_3$” based on reuse of logical network layer addresses. Such an inference is actually an inference of the type “a node with the address $a_1$ and a node with the address $a_2$ both communicated with a node with the address $a_3$”. Thus, reuse of a network layer identity on several occasions helps an attacker to resolve devices. In this simple example, the attacker discovers three devices – one device that stands apart by virtue of its popularity and two more that send to it. A node that uses several addresses with every other node is immune to such inferences, assuming that its multiple addresses cannot be linked to one another as belonging to the same device. In this simple example, the node $p_3$ could have used an address $a_{3,1}$ with node $p_1$ and an alternate address $a_{3,2}$ with node $p_2$.

6. A global eavesdropper with perfect spatial eavesdropping resolution cannot detect that a node has changed its address without auxiliary information telling him that two addresses belong to the same device. This is not because the change of address took place in a geographical “blind spot”, as could happen with a more local attacker, but rather because nodes move about, and two eavesdropped messages that originate from the same location do not necessarily belong to the same device. Hence, in order to detect a change of address, an attacker must keep track of nodes even when they are not transmitting (e.g., by following their movements closely, and later correlating origin of eavesdropped messages with node locations). If the attacker cannot do this, he may use movement prediction instead, but we note that prediction can be countered by introducing random periods of silence following every address change [87][63].

7. Linking packets as belonging to the same pair of unidentified devices does not hinder anonymity. In any case, since we will be assuming a constrained adversary, packet counting attacks [6] and countermeasures [50][38] are outside the scope.

5.2 Limitations

We limit the extent of our analysis due to constraints of scope. For the sake of argument, let us assume that operations outside the scope of this work do not hinder anonymity:

- Joining the network is an anonymous operation.
- Searching for a desired partner is an anonymous operation, resulting in the searching device’s acquisition of an address that is used for reception by its desired partner. Information about the searcher, its partner and their relationship is not revealed in the process. In particular, identity agreement algorithms operate anonymously.
- Upper (e.g., transport, application) layers are anonymous.
- Lower (e.g., datalink, physical) layers are anonymous.
- Leaving the network is an anonymous operation.
Although datalink layer topology affects emissions, potentially revealing topological node-dependent biases, our analysis ignores datalink layer topology, as it largely depends on the arrangement of nodes in spacetime. Regrettably, our analysis only pertains to cases where the attacker is blind to the effects of topology, to cases where the topology is highly symmetric, to cases where the topology is randomly chosen without node-dependent bias and to cases where a bias in the topology is masked by the algorithm (see Appendix C).

Lastly, as mentioned in section 2.1, this work focuses on the relation of unobservable communication patterns to observable addresses. As such, we analyze anonymity as affected by identities alone. Let us deliberately limit attacker strength accordingly. We assume a passive, external, global eavesdropper with zero spatial eavesdropping resolution, whose view contains only the addresses used by nodes, but not the execution’s delivery events, nor their time of occurrence. Given this, the attacker may count the messages containing a given address, but does not know the order in which they were sent or received. In addition, we assume that the attacker ignores all other message fields (e.g., flood identifiers).

5.3 Preliminaries

We present new definitions and notation required for our analysis. Recall $G$ as an input multigraph, $A$ as the address space and $GenID$ as an identity agreement algorithm. Let $\mathcal{K}(G) = E(G) \rightarrow A$ be the set of arc labelings of $G$ (i.e., identity agreement instances).

As it is probabilistic, $GenID$ induces a probability distribution over $\mathcal{K}(G)$. We define $E_{G \rightarrow \kappa}^{GenID}$ to be the event where $GenID$ computes a particular labeling, $\kappa \in \mathcal{K}(G)$. Such an event models the occurrence of a particular allocation. Since allocations are composite, so is such an event. Let $a \in A$ and $e \in E(G)$. We define $E_{e \rightarrow \kappa}^{GenID}$ to be the event where $GenID$ computes a labeling $\kappa$ for which $\kappa(e) = a$ (i.e., $GenID$ labels $e$ with $a$). Such an event models the occurrence of a particular assignment of $a$. With symbols, we formulate

$$E_{G \rightarrow \kappa}^{GenID} = \bigcap_{e \in E(G)} E_{e \rightarrow \kappa(e)}^{GenID}$$

In addition, by inducing a probability distribution over $\mathcal{K}(G)$, an arc labeling algorithm also induces an equivalence relation on $E(G)$, indicating atomically labeled arcs. Arcs in $E(G)$ are considered atomically labeled if they are identically labeled under every possible (non-zero probability) labeling. For example, in the case of an algorithm that always assigns a single address per node, all arcs ingress to a node are bundled in the same equivalence class. More formally, let us denote this equivalence with $\equiv$ and define it as follows:

$$\equiv \doteq \{(e_1, e_2) \in E(G)^2 \mid \forall \kappa \in \mathcal{K}(G) \quad Pr[E_{G \rightarrow \kappa}^{GenID}] > 0 \rightarrow \kappa(e_1) = \kappa(e_2)\}$$

We denote the resulting partition with $\mathcal{B}(G) = E(G)/\equiv$. One can think of the different equivalence classes as “bundles” of atomically labeled arcs. Since arcs in the same class are identically labeled, we overload our use of $\kappa$ accordingly, such that given $b \in \mathcal{B}(G)$,

$$E_{b \rightarrow \kappa(b)}^{GenID} = \bigcap_{e \in b} E_{e \rightarrow \kappa(b)}^{GenID}$$
Denote with $\mathcal{P}(X)$ the set of all multisets overlying the set $X$. We observe an attacker that can only view emitted destination indicators – one for whom $\mathcal{C}_A = \mathcal{P}(A)$ and whose discernible “colors” are multisets of destination indicators. With respect to a fixed $G$, we define algorithm emissions as a function of the agreement, $f_G : \mathcal{K}(G) \rightarrow \mathcal{C}_A$, where

$$f_G(\kappa) = \biguplus_{e \in E(G)} \{\kappa(e)\}$$

With words, the attacker views the sum total of aliases used in message deliveries. For instance, taking our example $G$ and $\kappa$ from section 3.2, we have $f_G(\kappa) = \{3, 14, 15, 92\}$. In reality, emissions depend on datalink layer topology as well – when messages to $p_j$ are relayed through others, its aliases are duplicated, appearing once per sent frame. Although we ignore the issue in this work, future work may wish to revisit the definition of $f_G$.

Let $K_{G \rightarrow c} = \{\kappa \in \mathcal{K}(G) \mid f_G(\kappa) = c\}$ be the set of arc labelings of $G$ that result in emissions observed as $c$. Substituting $G$ for $x$ in the notation of section 3.1, we have

$$E_{G \rightarrow c} = \bigcup_{\kappa \in K_{G \rightarrow c}} E_{G \rightarrow \kappa}^{GenID}$$
Chapter 6

Mononymous routing

In this chapter, we propose that existing algorithms achieve anonymity by making use of unbiased identity agreement algorithms. In an unbiased identity agreement algorithm, all nodes are equally likely to be assigned a particular address. Consequently, upon viewing a particular address, the attacker is unable to know which node is actually using it.

Below, we prove that the use of such an unbiased identity agreement algorithm results in anonymity. In section 6.1, we define two mononymous identity agreement algorithms that assign a single identity to each node. One algorithm assigns identities based on MAC addresses, in the spirit of existing allocation protocols in unadministered networks [17]. The other assigns identities in an unbiased manner. In sections 6.2 and 6.3, we analyze the anonymity of FRAs and DSR, respectively, when plugging in these agreement algorithms. Perhaps surprisingly, we are led to conclude that insofar as communication patterns are reflected in observable addresses, MANETs are already “inherently” anonymous.

Although one may naturally think of identities as being associated with nodes, we define lack of bias in an identity agreement algorithm in terms of arcs and their labels, rather than nodes and their addresses, for the sake of consistency with other definitions in this work.

Definition 4 Two multigraphs are isomorphic under GenID, denoted \( G_1 \simeq_{GenID} G_2 \), if there exists a bijection, \( \pi : B(G_1) \to B(G_2) \), such that for all \( b \in B(G_1) \), it holds that \( |b| = |\pi(b)| \). We denote the set of all such isomorphisms from \( G_1 \) to \( G_2 \) with \( \Pi^{GenID}_{G_1,G_2} \).

Definition 5 An arc labeling algorithm, GenID, is unbiased if it satisfies the following two conditions, describing properties of independence and equiprobability, respectively:

1. \( \forall G, \forall \kappa \in K(G), \) the events \( E_{b \rightarrow \kappa(b)}^{GenID} \), for \( b \in B(G) \), are mutually independent.

2. \( \forall G_1, \forall G_2, \) if \( G_1 \simeq_{GenID} G_2 \), then \( \forall b \in B(G_1), \forall \pi \in \Pi^{GenID}_{G_1,G_2} \) and \( \forall a \in A \),

\[
Pr[E_{b \rightarrow a}^{GenID}] = Pr[E_{\pi(b) \rightarrow a}^{GenID}]
\]

In terms of nodes and their addresses, the conditions above imply that addresses are assigned in an independent manner, with assignments revealing nothing about one another, and that for any given address, all of its possible assignments are equally likely.
6.1 Agreements

Let $h$ be a hash function. We make no requirements of $h$, except that it be deterministic. Let $Q: A \rightarrow [0, 1]$ be a probability distribution over the address space. For the rest of this chapter, let us consider two types of agreements, parametric in $h$ and $Q$, respectively:

**Definition 6 (Mononymous Agreements)**

$GenID_{\text{MAC-HASH}}^\text{PER-NODE}(h)$: labels all arcs to a node with a hash of its MAC address.

$GenID_{\text{UNBIASED}}^\text{PER-NODE}(Q)$: labels all arcs to a node with an address randomly drawn from $Q$. Addresses are independently drawn from the same $Q$, thereby ensuring lack of bias.

6.2 Mononymous FRAs

In this section, we analyze flood routing algorithms under the agreements defined above. A flood routing algorithm that uses an agreement algorithm $X$ is denoted with $FRA(X)$. Our results are presented in the form of annotated theorems. Theorems 1 and 2 show how the above agreements affect anonymity and Theorem 3 compares them to each other.

**Theorem 1** $FRA(GenID_{\text{MAC-HASH}}^\text{PER-NODE}(h))$ does not achieve observational anonymity.

**Proof** We demonstrate two inputs, $G_1, G_2$, such that $G_1 \simeq G_2$ but $G_1 \not\approx G_2$.

The two inputs are shown to the left. Clearly, $G_1 \simeq G_2$. Yet, we find that $G_1 \not\approx G_2$ because there exists $c = \{h(MAC(p_2))\}$, for which $Pr[E_{G_1 \leftarrow c}] = 1$, but $Pr[E_{G_2 \leftarrow c}] = 0$. ■

Theorem 1 confirms the intuition that uniquely tagging nodes has a negative impact on anonymity. As far as anonymity is concerned, an identity agreement which is based on unique information is, in some sense, the worst possible kind of identity agreement.

In contrast, an unbiased agreement does not rely on unique information and, therefore, achieves observational anonymity, as is shown in Theorem 2. More generally, Theorem 2 shows that isomorphism under an unbiased algorithm implies indistinguishability.

In order to understand Theorem 2, the reader is encouraged to refer to the analogy between anonymity and secrecy. According to the analogy, communication patterns act as “plaintext”, emitted addresses act as “ciphertext” and agreements telling which addresses are identities of which nodes act as secret “keys”. Thankfully, our notation endorses this view – $b \in B(G)$ represents a “bit” in the input, $c \in C_A$ represents a “ciphertext” and $\kappa \in K(G)$ represents a “key”. As with ciphers, input “bits” are combined with the “key” in order to produce a “ciphertext”. In Theorem 2, we prove that the “keys” given by an unbiased agreement algorithm perfectly mask isomorphic inputs. The intuition behind the proof is that addresses are semantically neutral values, used for the sole purpose of coordination. As such, it matters not which addresses are assigned as identities of which nodes and there exist multiple overlapping allocations that incur identical emissions.
Theorem 2 \( \text{FRA}(\text{GenID}_{\text{PER-NODE UNBIASED}}(Q)) \) achieves observational anonymity.

Proof Let \( G_1, G_2 \) be isomorphic inputs. Let \( c \in \mathcal{C}_A \). We show \( \Pr[E_{G_1 \rightarrow c}] = \Pr[E_{G_2 \rightarrow c}] \). For clarity, set \( \text{GenID} = \text{GenID}_{\text{PER-NODE UNBIASED}}(Q) \), \( B_i = B(G_i) \) and \( K_i = K_{G_i \rightarrow c} \) \((i = 1, 2)\).

We show that an unbiased agreement algorithm compensates for node permutation, in that addresses assigned to nodes are equally likely to have been assigned to their isomorphic counterparts, without a change in emissions. Trivially, \( G_1 \simeq G_2 \) implies \( G_1 \simeq_{\text{GenID}} G_2 \).

Given an alteration of the input, \( \pi \in \Pi_{G_1,G_2}^{\text{GenID}} \), let us observe a corresponding, emission-preserving, alteration of the agreement, \( \delta: K_1 \rightarrow K_2 \), defined in terms of \( \pi \) as follows:

\[
\delta = \{(\kappa_1, \kappa_2) \in K_1 \times K_2 \mid \forall b \in B_1 \quad \kappa_1(b) = \kappa_2(\pi(b))\}
\]

By definition, \( \delta \) is bijective. Hence, so is \( \delta \times \pi \). Since \( \text{GenID} \) is unbiased, \( \Pr[E_{G_1 \rightarrow c}] \) and \( \Pr[E_{G_2 \rightarrow c}] \) are both expressible as sums of probability products, where \( \delta \times \pi \) gives a probability-preserving mapping between constituent events, giving the desired equality:

\[
\Pr[E_{G_1 \rightarrow c}] = \Pr\left[ \bigcup_{\kappa_1 \in K_1} E_{G_1 \rightarrow \kappa_1}^{\text{GenID}} \right] \quad \text{(by definition)}
\]

\[
= \sum_{\kappa_1 \in K_1} \Pr[E_{G_1 \rightarrow \kappa_1}^{\text{GenID}}] \quad \text{(by disjointedness)}
\]

\[
= \sum_{\kappa_1 \in K_1} \Pr\left[ \bigcap_{b_1 \in B_1} E_{b_1 \rightarrow \kappa_1}^{\text{GenID}}(b_1) \right] \quad \text{(by associativity)}
\]

\[
= \sum_{\kappa_1 \in K_1} \prod_{b_1 \in B_1} \Pr[E_{b_1 \rightarrow \kappa_1}^{\text{GenID}}(b_1)] \quad \text{(by independence)}
\]

\[
= \sum_{\kappa_2 \in K_2} \prod_{b_2 \in B_2} \Pr[E_{b_2 \rightarrow \kappa_2}^{\text{GenID}}(b_2)] \quad \text{(by equiprobability)}
\]

\[
= \sum_{\kappa_2 \in K_2} \Pr\left[ \bigcap_{b_2 \in B_2} E_{b_2 \rightarrow \kappa_2}^{\text{GenID}}(b_2) \right] \quad \text{(by independence)}
\]

\[
= \sum_{\kappa_2 \in K_2} \Pr[E_{G_2 \rightarrow \kappa_2}^{\text{GenID}}] \quad \text{(by associativity)}
\]

\[
= \Pr\left[ \bigcup_{\kappa_2 \in K_2} E_{G_2 \rightarrow \kappa_2}^{\text{GenID}} \right] \quad \text{(by disjointedness)}
\]

\[
= \Pr[E_{G_2 \rightarrow c}] \quad \text{(by definition)}
\]

\[\blacksquare\]
For the sake of visual demonstration, we apply Theorem 2 to our example input from section 3.2. Below, our old example input is shown on the left. On the right, we show an isomorphic input that is obtained by application of a permutation, say $(p_1)(p_2, p_3)$:

Under a per-node agreement, the two inputs are partitioned into “bits” as follows:

A brief glance reveals isomorphism under the agreement algorithm, with isomorphic “bits” identically circumscribed. The partition dictates the structure of “ciphertexts” and “keys”.

“Ciphertexts” are of the form $c = \{a_i, a_j, a_j, a_j\}$ and generated “keys” look as follows:

Theorem 2 shows that the above two “keys” are equiprobable, despite the permutation. In an unbiased agreement algorithm, the equiprobability property ensures that isomorphic “bits” are equally likely to be equally labeled, whereas the independence property ensures that local equiprobability extends to the entire multigraph. In the above, $p_2$ and $p_3$ are equally likely to own either $a_i$ or $a_j$, thereby securing their anonymity. Informally speaking, nodes are permuted “from underneath” the agreement, without affecting probability.

Above, we have shown that biased agreements are not on par with unbiased ones. But, the fact that an unbiased agreement succeeds where a biased one fails does not comprise a formal comparison of their relative strength. This is rectified in Theorem 3, below.

**Theorem 3** $\text{FRA}(\text{GenID}^{\text{PER-NODE-MAC-HASH}}(h)) \leq \text{FRA}(\text{GenID}^{\text{UNBIASED}}(Q))$.

**Proof** For clarity, set $\text{ALG}_i = \text{FRA}(\text{GenID}_i)$ for $i = 1, 2$ and prove that $\text{ALG}_1 \leq \text{ALG}_2$. Let $G_1, G_2$ be inputs, such that $G_1 \sim_{\text{ALG}_1} G_2$. We show $G_1 \sim_{\text{ALG}_2} G_2$.

Since $\text{GenID}_1$ computes hashes of predetermined quantities, it is deterministic. Let us denote the labeling of $G$ with $\kappa_G$. Since $f_G$ is also deterministic, $\kappa_G$ uniquely determines emissions, $c_G = f_G(\kappa_G)$. Since $G_1 \sim_{\text{ALG}_1} G_2$, executions on $G_1$ and $G_2$ result in the same emissions, $c_{G_1} = c_{G_2}$. Assuming there are no collisions, the well-defined $\kappa_{G_2}^{-1} \circ \kappa_{G_1}$ shows that $G_1 \simeq_{\text{GenID}_2} G_2$. Since $\text{GenID}_2$ is unbiased, the result follows (by Theorem 2).

As a corollary, $\text{FRA}(\text{GenID}^{\text{PER-NODE-MAC-HASH}}(h)) < \text{FRA}(\text{GenID}^{\text{UNBIASED}}(Q))$. However, despite its apparent inferiority, an algorithm that hashes MAC addresses can still achieve observational anonymity by simply including unbiased randomness in the computation. For instance, an algorithm may use unbiased nonces or, instead, unbiased MAC addresses to begin with; with all traces of bias removed from the agreement, Theorem 2 applies.
6.3 Mononymous DSR

Since the phases of DSR can each be modeled as a variation of flood routing, we conjecture that an analogous analysis yields analogous results, whereby a symmetric routing algorithm that uses an unbiased identity agreement algorithm achieves observational anonymity. In general, since the properties of FRAs were not strictly required in the analysis, it is possible that the same applies to any algorithm that uses the abstraction of an identity.
Chapter 7

Polyonymous routing

In this chapter, we propose to increase information hiding, by letting nodes use multiple network layer addresses to communicate with one another. These can include alternate addresses for transmission and reception, alternate addresses for different senders, alternate addresses for routing on different routes and alternate addresses per transferred packet.

As shown in the previous chapter, anonymity is practically “free”, in that it only requires a change in agreements and not in routing algorithm procedures. However, anonymity is not enough to protect nodes from harm. Even in the case of unbiased agreements, where an address does not directly lead back to its owner, an attacker that learns an address can use it to compromise its owner, as described in section 2.4.1. In order to make such attacks more costly, we wish to prevent an attacker from narrowing candidates for attack.

Despite their observational anonymity, the algorithms of the previous chapter divulge much of the structure of their input in their emissions, such that inputs with different structures are not mapped to the same observational equivalence classes, let alone find themselves together in $\sim$. If attack candidacy is somehow related to input structure, an attacker can use this fact to his advantage. For example, an attacker can observe emissions, select the most frequently observed address and concentrate on attacking its owner$^1$.

In order to combat such attacks, let us hide the inherent structural features of the input by additional symmetrization. In the algorithms presented above, receivers were assigned a single address, such that for any pair of different target values, $t_1 \in T_{i_1}$ and $t_2 \in T_{i_2}$, an equality of the associated receivers implied an equality of used destination indicators, $\text{Encode-Destination-Indicator}_{i_1}(t_1)$ and $\text{Encode-Destination-Indicator}_{i_2}(t_2)$. Instead, letting a receiver use multiple identities with different senders breaks this implication, for $i_1 \neq i_2$. By letting the receiver do so with the same sender, we break it for $i_1 = i_2$ as well$^2$.

Below, we prove that a plurality of identities aids information hiding without retarding anonymity. In section 7.1, we define two polyonymous identity agreement algorithms that assign multiple identities to every node. One algorithm assigns identities per channel (per node pair). The other assigns identities per sent message (per arc). In sections 7.2 and 7.3, we augment the results of the previous chapter and conclude our formal analysis.

---

$^1$By comparison, in human-based MANETs, examples of structure-based targeting include presidential assassinations, the flocking of paparazzi and the deeds of Robin Hood and his band of Merry Men.

$^2$Interestingly, a node might know the same partner by multiple identities without realizing it.
7.1 Agreements

Let us consider two types of agreements:

Definition 7 (Polyonymous Agreements)

GenID$^{PER-CHAN}_{UNBIASED}(Q)$: labels all arcs of a node pair with an address randomly drawn from $Q$. Again, addresses are independently drawn from the same $Q$ to avoid bias.

GenID$^{PER-SEND}_{UNBIASED}(Q)$: labels individual arcs with addresses randomly drawn from $Q$. Were we to regard time, nodes could be thought of as “hopping” between addresses.

7.2 Polyonymous FRAs

We continue our analysis in the style of the previous chapter. Since the new agreements are unbiased, both achieve anonymity; the proof is identical to that of Theorem 2. Theorems 4 and 5 show that per-channel agreements are not comparable to per-node agreements. Afterwards, Theorems 6 and 7 show that per-message agreements are better than both.

A per-channel agreement conceals the number of receivers in the input, but the number of channels is revealed. Hence, while not always better, per-channel agreements do succeed in important cases where per-channel agreements fail. In Theorem 4, let us juxtapose inputs possessing the same number of channels but with a different number of receivers.

Theorem 4 $FRA(GenID^{PER-CHAN}_{UNBIASED}(Q_x)) \not\leq FRA(GenID^{PER-NODE}_{UNBIASED}(Q_y))$.

Proof For clarity, set $ALG_i = FRA(GenID_i)$ for $i = 1, 2$ and prove that $ALG_1 \not\leq ALG_2$. We demonstrate two inputs, $G_1, G_2$, such that $G_1 \sim_{ALG_1} G_2$ but $G_1 \sim_{ALG_2} G_2$.

![Diagram](image)

The two inputs are shown to the left. Since $G_1 \sim_{GenID_2} G_2$ and $GenID_2$ is unbiased, $G_1 \sim_{ALG_2} G_2$. Observe $c = \{a_i, a_j\}$, where $a_i \neq a_j$. Since $G_2$ has one channel, $Pr[E^{GenID_1}_{G_2 \rightarrow c}] = 0$. Yet, since $G_1$ has two channels, $Pr[E^{GenID_1}_{G_1 \rightarrow c}] > 0$, implying our result.

A per-node agreement conceals the number of channels in the input, but the number of receivers is revealed. Hence, while not always better, per-channel agreements do succeed in important cases where per-node agreements fail. In Theorem 5, let us juxtapose inputs possessing the same number of channels but with a different number of receivers.

Theorem 5 $FRA(GenID^{PER-CHAN}_{UNBIASED}(Q_x)) \not\leq FRA(GenID^{PER-NODE}_{UNBIASED}(Q_y))$.

Proof For clarity, set $ALG_i = FRA(GenID_i)$ for $i = 1, 2$ and prove that $ALG_1 \not\leq ALG_2$. We demonstrate two inputs, $G_1, G_2$, such that $G_1 \sim_{ALG_1} G_2$ but $G_1 \sim_{ALG_2} G_2$.

![Diagram](image)

The two inputs are shown to the left. Since $G_1 \sim_{GenID_2} G_2$ and $GenID_1$ is unbiased, $G_1 \sim_{ALG_1} G_2$. Observe $c = \{a_i, a_j\}$, where $a_i \neq a_j$. Since $G_1$ has one receiver, $Pr[E^{GenID_1}_{G_1 \rightarrow c}] = 0$. Yet, since $G_2$ has two receivers, $Pr[E^{GenID_2}_{G_2 \rightarrow c}] > 0$, implying our result.
As demonstrated above, by naming channels instead of nodes, popular receivers with numerous senders cannot be told apart from less popular ones, and candidates for attack cannot be eliminated based on structural features of the input. Inputs having a different number of receivers but the same number of channels (with matching volumes) become indistinguishable, bringing us a step closer towards node-count immeasurability.

As a natural extension of per-channel agreements, it is also possible to name individual transmissions. For example, consider a scenario in which each message contains the address to be used (e.g., by agreeing in advance on a complete list\(^3\), by agreeing in advance on a shared seed and using it to compute random addresses in a synchronized fashion, or by using a scheme similar to one-time passwords \([41][42]\)).

We agree in advance on the addresses to be used (e.g., by agreeing in advance on a complete

As demonstrated above, by naming each message, popular channels or receivers cannot be told apart from less popular ones. Inputs having the same volume of messages become indistinguishable, bringing us a step closer towards node-count immeasurability.

\(^3\)As an example of similar tactics outside the context of MANETs, consider the Mossad operatives who (with a late relative of the author’s among them) captured and brought Eichmann to justice. They would rendezvous on multiple occasions at different pre-agreed upon times and cafés in Buenos Aires \([43]\).
7.3 Polyonymous DSR

Once again, we conjecture analogous results for DSR, whereby more freedom in the selection of addresses preserves anonymity, while rendering more inputs indistinguishable.

In addition, we propose that a router wishing to remain anonymous use multiple router indicators for different routes, such that the router indicators used by a single router cannot be related to one another. Otherwise, a router can be shown to have participated in two separate routes, making DSR susceptible to a logical neighborhood correlation attack. With the assumption that logical routes represent the shortest path possible in the datalink topology graph, this can indicate topological relationships between route endpoints (i.e., two logical routes containing many identical router indicators run through close neighborhoods of the datalink topology graph). With the assumption that datalink topology is governed by physical topology, this can indicate spatial relationships between nodes.

In order to account for this, only a slight modification to DSR-Encode-Router-Indicator is required. In the modified version, instead of using a constant router indicator, a router randomly draws, uses and stores a new router indicator per encountered route request.

As long as router indicators are not changed mid-session, DSR’s local repair optimization remains viable, as it relies on their constancy throughout the session. When mid-session changes do occur, enabling local repair is not trivial, and may require sophisticated synchronization mechanisms. Additionally, under polyonymous routing, DSR’s route caching optimization becomes useless, because it relies on nodes’ ability to share knowledge about destinations. In some sense, a node is forced to “work alone” when discovering routes in its environment, because each node has its own “world view” of destination identities.

Given knowledge about transmission ranges, observed route lengths can indicate physical properties of the network, such as an upper bound on its physical diameter. In DSR, route length is immediately indicated by an observed source route field. We conjecture that a low cost method for preventing inferences based on observed route length is to have nodes randomly choose to play the role of (possibly multiple) virtual nodes.
Chapter 8

Anonymity-aware systems

In this chapter, we continue our investigation with three existing routing algorithms that were designed with anonymity in mind. Our motivation in examining anonymity-aware systems is to be able to compare and contrast our approach with existing ones. We describe the three algorithms in an informal manner, in sections 8.1, 8.2 and 8.3, respectively\(^1\).

A complete and detailed survey is outside the scope of this work, but the interested reader is encouraged to refer to related work, cited in chapter 10. A comparison of all the algorithms in this work, including those of this chapter, awaits the reader in chapter 9.

\(^{1}\)With apologies to their original authors for any omissions or ill-described features.
8.1 Mix Route Algorithm

Jiang et al. propose a direct use of mixes in a MANET [49]. In their design, designated nodes in the MANET serve as mixes (“mix nodes”) and offer anonymity services to regular nodes (“non-mix nodes”) that register with them. The mix nodes form an overlay and the underlying routing algorithm of the MANET is used for routing between them.

8.1.1 Assumptions

Network

- Nodes use long-lived addresses for routing.
- Addresses contain no location information, but may disclose mobile user identity.
- There exists an underlying routing algorithm in the MANET.

Adversary

- The system tries to prevent an attacker from performing traffic analysis.
- The probability that a roaming node is captured is considered, and the attack model specified is one of a global eavesdropper that is bounded in computing power and bounded in node intrusion capability (no more than $K$ mixes in a time window $T$).
- The system tries to defend against both internal and external attackers. The authors define an external attacker to be one who is thwarted by mixing approaches and cannot learn the relationship between mix inputs and outputs. An internal attacker is one who has compromised a mix. Given such an attacker, anonymity relies on the existence of at least one trusted mix in the path between sender and receiver.
- The network does not possess perfect intrusion detection.

Trust

- A PKI is required in order to enable public key onioning in mix-net designs.
- A sender must know the receiver’s public key in order to send a message to it.
- A mutual authentication mechanism is needed for the non-mix nodes to establish trust relationships with mix nodes that advertise the anonymity service.
- When selecting a mix route between sender and a receiver, a random set of mixes is chosen. This entails the existence of some knowledge about the mixes that are members in the network, and a knowledge of their public keys.
8.1.2 Specification

The mix nodes are discoverable via the use of proactive mix advertisements, which are propagated across the network in a controlled fashion. Non-mix nodes each maintain an entry pointing to the closest mix node ("dominator mix"). If a non-mix node has not heard from its recorded dominator mix for a while, it replaces it with the next closest one.

A sender attempting to contact a receiver first chooses a mix route upon which to forward the contact request. The chosen mix route consists of a random set of mix nodes (which can include the sender’s dominator mix), and is later adjusted to improve performance.

The sender then issues an RREQ across the chosen mix route\(^2\). The RREQ, like other messages traversing a mix route, is onioned. Onioning ensures that each mix node in the mix route knows only the identities of the previous and next mix nodes in the mix route.

Once the receiver has been contacted, it registers with the last mix in the route. Then, in the data transfer phase, a sender can send onioned messages, such that the identities of participating mixes are hidden and mixes’ input-output relationships are scrambled.

Like advertisement messages, registration messages are periodically resent in order to maintain accurate records despite node motion. A route update mechanism is also provided in order to improve upon the initial mix route selection.

---

\(^2\)Jiang et al. explain that the RREQ is a unicast message, and as such can be encrypted with the receiver’s public key, such that an attacker cannot trace the message. However, an attacker capable of low level flow analysis can observe the RREQ arriving at a receiver without leaving it again, and realize that a sender is trying to establish a route to it. Silent reception can prevent this attack (see Appendix A).


8.2 Secure Distributed Anonymous Routing (SDAR)

SDAR [12] is an on-demand onion-routing-based routing protocol for MANETs. In terms of routing, SDAR is similar to AODV [75], in that floods serve to build distributed state.

8.2.1 Assumptions

Network

- Nodes use single long-lived identifiers for routing.
- Nodes do not use an underlying routing protocol.

Adversary

- An adversary can compromise nodes.
- A variety of techniques can be used to infer node identities, locations and relationships.
- Malicious actions include affecting message coding, timing, message volume, flooding, intersection and collusion.

Trust

- A PKI is required for ensuring that a sender knows the real public key of a receiver.
- A trust management system is defined, in which trust is cumulative and is based on previous node behavior (message modification\dropping history of a node)\(^3\).
- A node in the MANET belongs to one of three trust communities – no trust, medium trust and high trust. The initial trust level of a node is established upon registration. All nodes in the medium trust community share a single secret symmetric key. All nodes in the high trust community also share a (different) secret symmetric key.
- Symmetric keys are delivered to neighbors in a neighborhood by a central node of the neighborhood (encrypted in separate under the nodes’ published public keys). When a neighbor leaves or when its trust level is reduced, the keys it had become disqualified and new keys are exchanged between the remaining neighbors.

\(^3\)It would seem that basing trust on previous behavior could be problematic, because such behavior is not necessarily measurable in MANETs and because an attacker can pretend to be a legitimate node for as long as necessary, before deviating from “normal” behavior and performing an attack.
8.2.2 Specification

At first, a sender floods an RREQ. All nodes receiving the RREQ perform a public key operation, in order to determine whether they are the intended recipient or not. If a node receiving the RREQ is not the intended recipient, it adds its own identifying information to it, in encrypted form, and notes the predecessor node in a table. Nodes along the way do not simply broadcast the RREQ as they would in an FRA, but instead only forward it to neighboring nodes that meet the trust requirements specified by the sender. In this manner, routes are guaranteed to consist of nodes with a certain minimum level of trust.

Once the intended receiver is found, it reconstructs the return path using the information it received, and issues an RREP to the sender, along this path. The RREP contains the identities and symmetric keys known by the nodes in the path, which is all the information necessary for the sender to send future data across the path. As the RREP travels backwards towards the sender, nodes along the way peel the encrypted layers within it and record successor node IDs in a table, alongside the previously recorded predecessor IDs.

In the data transfer phase, the sender iteratively encrypts the message to be sent, using the symmetric keys obtained from the RREP. The onioned message is then sent across the network, with each node in the path peeling its own layer, in turn, until the message arrives at the receiver, who peels the last layer to reveal the contents of the message.

Lastly, in terms of managing communities, SDAR employs a proactive approach. Nodes broadcast periodic HELLO messages to help nodes keep track of their neighbors. HELLO messages contain a node’s public key, which is stored by its neighbors when encountered for the first time. A mechanism for removing stale key entries is also provided.
8.3 Anonymous Routing Protocol for MANETs (ARM)

ARM [91] is an on-demand onion-routing-based routing protocol for MANETs. In terms of routing, ARM is also similar to AODV. ARM addresses several deficiencies of SDAR. First, in the RREQ phase, it minimizes the amount of public key cryptography required. When receiving a flood, a node performs a table lookup instead of a public key operation. Second, during data transfer, it distributes cryptographic workload among nodes. When preparing a message, the sender encrypts it once instead of onioning it (which would require $k$ encryptions given path length $k$). Then, routers re-encrypt the message per link.

8.3.1 Assumptions

Network

- Every node in the network has a permanent identity that is known by the other nodes in the network that wish to communicate with this node.
- Wireless links between nodes are symmetric.

Adversary

- The first adversary considered is an external global passive adversary who can observe all possible communications between all nodes in the network at all time.
- The second adversary considered is a cooperating node inside the network, which means that every node in the network is potentially colluding with the adversary.

Trust

- There is no need for nodes to have public keys initially known by others.
- A source node $S$ and a destination node $D$ share a secret symmetric key, $k_{SD}$ and a secret pseudonym, $Nym_{SD}$. The authors do not specify how the nodes came to agree upon these, but assume that the agreement was made in advance.
- Nodes only share secret keys and pseudonyms with a limited number of other nodes.
- Every node anonymously establishes a broadcast key with its 1-hop neighborhood.
8.3.2 Specification

In order to combat flow analysis attacks during route discovery, which are not addressed by SDAR, the authors of ARM ensure that an RREQ propagates across the entire network, even after it has reached its target. An RREP does the same, such that the real sender to which it is destined is hidden. Loosely speaking, thwarting flow analysis in ARM relies on RREQs, RREPs and data appearing random and flowing everywhere at all times.\(^4\)

To find \(D\), \(S\) generates a new asymmetric key pair, \(pub_D/priv_D\), and a symmetric key, \(k\). In the RREQ, \(S\) encrypts \(priv_D\), \(k\) and \(D\)'s permanent identity under \(k_{SD}\). This ensures secrecy of \(priv_D\) and \(k\) and allows \(D\) to authenticate \(S\). \(S\) also attaches an authenticator to be used by path members to verify that an RREP truly originates from \(D\). Then, \(S\) generates a random link identifier of the form \((n_S, k_S)\) and adds it to an asymmetric onion encrypted under \(pub_D\). Lastly, \(S\) attaches \(Nym_{SD}\), a \(ttl\) value and \(pub_D\) to the RREQ. \(Nym_{SD}\) doubles as a flood identifier, allowing superfluous RREQs to be discarded.

A node receiving the RREQ checks whether it owns \(Nym_{SD}\). If not, then it is a relay. A relay generates a new link identifier, \((n_i, k_i)\), and adds it to the asymmetric link identifier onion. Before forwarding the RREQ, it records \((Nym_{SD}, n_i, k_i, auth)\) in its routing table, where \(auth\) is the authenticator. Otherwise, when it is \(D\) that receives the RREQ, it uses \(k_{SD}\) to extract \(priv_D\) and uses \(priv_D\) to peel the asymmetric link identifier onion.

Using the link identifiers inside, \(D\) constructs a symmetric reply onion. Each node's layer is encrypted under its \(k_i\) and contains its \(n_i\) as well as a shared key to use with its upstream neighbor and information to validate the authenticator. Then, the RREP is sent across the network. A node receiving the RREP, and that has recently forwarded a matching RREQ, learns if it is part of the return path by decrypting the symmetric onion with its \(k_i\) and looking for its \(n_i\) in the result. If so, it forwards the RREP with one layer peeled. If not, it still forwards the RREP, but replaces the onion with random data. In any case, the RREP header re-encrypted under different broadcast keys at every hop. When the process ends, every pair of neighbors on the return path shares a secret key.

When \(S\) sends a data message, it attaches a one-time identifier to it, that is computed using the next hop key. The computation is simply an encryption of an incrementing counter under the key. Whenever a data message is delivered from one hop to the next, both sides increment the link counter. The payload itself is encrypted under this same symmetric key. As with RREPs, the header is encrypted under the neighborhood broadcast key.

When a node receives a data message with an identifier, it determines whether it is part of the path by looking up the identifier in its routing table. As a result of the lookup, the node finds the information about the next hop (the secret key that it shares with the next hop and the new identifier that it needs to place on the data message). If it is indeed part of the path, then the payload is decrypted, and then re-encrypted using the secret key that is shared with the next hop. The new identifier is placed on the data message and the message is sent to the next hop router. Nodes that are not part of the path replace the payload and message identifier with random data and continue relaying the message.

\(^4\)However, despite these measures, a global attacker capable of performing a sufficiently precise flow analysis attack can realize the origin of RREQs, since an originating node generates an output without having any inputs. This attack cannot be averted, unless a cover traffic scheme ensures that dummy RREQs are simultaneously and symmetrically being originated in multiple other locations in the network.
Chapter 9

Comparison

In this chapter, we draw a comparison of the algorithms presented in this work. We note that none is necessarily superior to any other – the different algorithms are suitable for different scenarios and pursue different tradeoffs between cost and delivered guarantees.

The algorithms of the previous chapter implement routing algorithm procedures with symmetric and asymmetric cryptographic primitives judiciously sprinkled throughout the route setup and data transfer phases. In the case of the mix route algorithm and SDAR, a PKI must be maintained in order to support these operations. In the case of ARM, nodes must establish secret pairwise key agreements and neighborhood keys. In both cases, cryptography aims to provide several distinct guarantees, hopefully offsetting its cost.

With cryptography, we are afforded the following guarantees against a computationally bounded, external, adversary. First, participants can assure message secrecy and integrity. Second, they can authenticate one another (except in the mix route algorithm, which leaves this item for future work). Third, random data is indistinguishable from encrypted data and can be used for cover. Fourth, a message can undergo metamorphosis en route, such that outer appearances cannot be used to link different sightings to one another. This is known as bitwise unlinkability. Without it, mixing is not possible. Without mixing, one cannot thwart a high resolution adversary. In the case of the mix route algorithm, mixing is explicit. In the case of SDAR, mixing is implicit. In the case of ARM, mixing is implicit by default, but the authors acknowledge its applicability as a general technique.

As with our proposed approach, the end goal is to allow communication to take place, while arriving at a transcript that is unintelligible, with emissions that reveal nothing about the communicating parties in the system (i.e., who communicates with whom). The authors of the algorithms of the previous chapter all hint at the need to have the values chosen as identities be known by as few entities as possible. In contrast, unbiased identity agreement algorithms make it possible for the values chosen as identities to be used in the clear, with the transcript acting as a “ciphertext”. In terms of the anonymity that is achieved against the weak adversary we considered, the end result is the same, with the slight exception that our proposal achieves unconditional security as opposed to computational security.

Of course, our proposal does not provide additional guarantees, nor protect against a stronger adversary. The former is by design – we aim for a clear separation of concerns and treat anonymity only. The latter is a matter left for future study. Notwithstanding, our proposal is accompanied by a rigorous formal analysis of the anonymity achieved.
Since the algorithms considered in this work make different assumptions about the problem of anonymity and provide different guarantees, we do not unfairly compare their anonymity towards a powerful adversary or their performance in terms of cryptographic overhead. Instead, let us compare algorithms in terms of the assumptions made about node-dependent biases and in terms of whether these biases are compensated for. Such a comparison is beneficial as it applies across the board and offers a general overview. In addition, it serves to focus our attention on the key issue of node-dependent bias.

We refer to the three algorithms of the previous chapter as MIX, SDAR and ARM. We refer to the FRAs analyzed in chapters 6 and 7 as FRA\textsubscript{i}, where \(i = 1, 2, 3, 4\) indicates the identity agreement algorithm considered, in order of their presentation. Since DSR is repeatedly conjectured to resemble FRAs, it is omitted from the comparison entirely.

The comparison is shown in Table 9.1. In the table, columns represent algorithms and rows represent node-dependent biases. For each algorithm and bias, we indicate whether the bias is assumed and whether it is compensated for. If the bias is assumed, we write “A”. Otherwise, we write “¬A”. If the bias is compensated for, we write “C”. Otherwise, we write “¬C”. In case of ambiguity, we write “?”. Table entries are explained below.

In the first row, we ask about node-dependent bias in the datalink layer topology. For instance, a node may be at a more central location than others, causing it to relay more traffic. As mentioned in section 5.2, our analysis ignores datalink layer topology. Hence, the algorithms proposed in this work are said to neither assume the existence of bias nor compensate for it. Algorithms proposed elsewhere do not explicitly refer to the issue.

In the second row, we ask about node-dependent bias in routing algorithm procedures. Since all of the algorithms presented in this work are symmetric, all are said to not assume the existence of such bias. Consequently, none of them need to compensate for it.

In the third row, we ask about node-dependent bias in the identity agreement. As FRA\textsubscript{1} relies on potentially biased MAC addresses, it is said to assume such bias. Disparagingly, it makes no compensation. In contrast, as FRA\textsubscript{2}, FRA\textsubscript{3} and FRA\textsubscript{4} guarantee lack of bias, they are said to assume none and, rejoicingly, obviate the need for compensation. Lastly, as MIX, SDAR and ARM hint at the existence of identities that must be hidden, they are said to assume bias. Fortunately, they compensate since addresses are concealed.

In the fourth row, we ask about node-dependent bias in communication structure. For instance, a node may be more popular than others, as discussed in chapter 7. Algorithms proposed in this work are said to assume such bias. Algorithms proposed elsewhere do not explicitly refer to the issue. Sadly, the mononymous FRA\textsubscript{1} and FRA\textsubscript{2} do not compensate for the bias. Happily, the polyonymous FRA\textsubscript{3} and FRA\textsubscript{4} do so, by hiding input structure. Lastly, as MIX, SDAR and ARM ensure completely random-looking transcripts, input structure is unlikely deducible at all, implying that bias is indeed compensated for.

In the fifth row, we ask about node-dependent bias in message contents. For instance, a node may be more likely to send a certain message than others are. Or, conversely, if every message is equally likely to be sent by everyone, then such a bias does not exist. The algorithms proposed in this work neither assume the existence of such bias nor attempt to compensate for it (i.e., message contents are completely ignored, as noted in section 2.3). Algorithms proposed elsewhere do not explicitly presuppose the existence of contents that must be hidden. However, they compensate since message contents are concealed.
The astute reader will notice numerous additional potential biases omitted from the table. For instance, one might consider a bias in message ordering, a bias in location or motion patterns, a bias in battery strength, a bias in produced audiovisual cues, a bias in software configuration (e.g., installed operating system, drivers, services, random number generators, statistics gathered at runtime et cetera) or a bias in hardware configuration (e.g., processor speed, number of cores, memory capacity, interface types et cetera). Since all such biases are considered to be outside the scope, they are purposely omitted.

<table>
<thead>
<tr>
<th>Bias in...</th>
<th>$FRA_1$</th>
<th>$FRA_2$</th>
<th>$FRA_3$</th>
<th>$FRA_4$</th>
<th>MIX</th>
<th>SDAR</th>
<th>ARM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datalink topology</td>
<td>$\sim A, \sim C$</td>
<td>$\sim A, \sim C$</td>
<td>$\sim A, \sim C$</td>
<td>$\sim A, \sim C$</td>
<td>$\sim ? , \sim ?$</td>
<td>$\sim ?, \sim ?$</td>
<td>$\sim ?, \sim ?$</td>
</tr>
<tr>
<td>Routing procedures</td>
<td>$\sim A, \sim C$</td>
<td>$\sim A, \sim C$</td>
<td>$\sim A, \sim C$</td>
<td>$\sim A, \sim C$</td>
<td>$\sim A, \sim C$</td>
<td>$\sim A, \sim C$</td>
<td>$\sim A, \sim C$</td>
</tr>
<tr>
<td>Chosen identities</td>
<td>$A, \sim C$</td>
<td>$\sim A, \sim C$</td>
<td>$\sim A, \sim C$</td>
<td>$\sim A, \sim C$</td>
<td>$A, C$</td>
<td>$A, C$</td>
<td>$A, C$</td>
</tr>
<tr>
<td>Comm. structure</td>
<td>$A, \sim C$</td>
<td>$A, \sim C$</td>
<td>$A, C$</td>
<td>$A, C$</td>
<td>$\sim ?, \sim ?$</td>
<td>$\sim ?, \sim ?$</td>
<td>$\sim ?, \sim ?$</td>
</tr>
<tr>
<td>Message contents</td>
<td>$\sim A, \sim C$</td>
<td>$\sim A, \sim C$</td>
<td>$\sim A, \sim C$</td>
<td>$\sim A, \sim C$</td>
<td>$\sim ?, \sim ?$</td>
<td>$\sim ?, \sim ?$</td>
<td>$\sim ?, \sim ?$</td>
</tr>
</tbody>
</table>

Table 9.1: Comparison of routing algorithms in this work
Chapter 10

Related work

In this chapter, we place our contributions in the context of related work. Since anonymity in MANETs borrows from wired anonymity, we describe related work in both fields.

10.1 Anonymity in wired networks

Anonymity, as it pertains to email delivery, is first considered by Chaum [15]. Chaum’s mix relays messages from users such that external observers are unable to determine which user’s messages are relayed to which other user, nor message contents. Messages are encrypted for the mix using its public key and, following decryption, are lexicographically reordered and retransmitted them towards their destination. In a mix cascade, or a series of mixes, messages are iteratively encrypted by the sender in advance and are opened by each mix in the series in turn. Chaum separately addresses the problem of anonymous broadcast (see [16] and its generalization in [26]). In contrast with Chaum, we do not assume that an attacker can pinpoint the origin and destination device of every message in the system.

A more recent realization of onion encryption is the onion routing approach [37]. Onion routing has been adopted and implemented in low latency anonymity networks such as Tor [24], in which a user relies on a central network core of trusted onion routers and a PKI, and includes routing instructions inside onion-encrypted messages. Onion routing has also been shown to be possible without a PKI, by Katti et al. [54], who use information slicing in order to hide routing information. In contrast with onion routing, by assuming unbiased identity agreements, our approach circumvents the need to hide routing information. Since we do not assume that identities carry a semantic charge, we do not hide them.

An alternative approach in wired network anonymity has been to forego a central network core, and instead spread the work and culpability among all participating anonymity network members. In Tarzan [32], network nodes create pairwise protected links, allowing the sender to create a chain of encrypted links that ends at the receiver’s side. In Crowds [84], network nodes use biased coin tosses to create a random path in the network graph, through which a sender’s message is routed to its destination. As with the use of chained proxies, these solutions hide one identity behind another, while minimizing each node’s knowledge of the identities of other involved nodes. This approach resembles that of rebel organizations, whose structure minimizes members’ ability to implicate other members, and
is intended to spread and dissolve overall culpability. In contrast, we do not require users
to join a crowd, but rather allow them to use the network regularly, so that regular nodes
can serve both for relaying transmissions and for spreading culpability (for cover)\textsuperscript{1}.

In all of the above schemes, it is either assumed that the network topology is mostly
static, that the sender is aware of the network topology in advance, that the routing of
packets between any two nodes in the network is a same-cost operation or that locating a
random node in the network is an inexpensive task. Unfortunately, these assumptions do
not hold in MANETs, and so direct porting of these schemes is not always viable [59].

\section{Anonymity in MANETs}

Aside from those described in chapter 8, relatively few approaches exist.

MASK [98] uses a proactive approach to establish anonymous pairwise links between
neighboring nodes in such a way that a node may transmit a packet with a source MAC
address of all 1’s, but still reach its unique, intended, neighbor. Once these links have been
set up, a sender can broadcast an on-demand route request that is propagated throughout
the network, until it reaches the intended recipient. Once a path has been established,
communication proceeds between sender and recipient using a virtual circuit switching
process. ALARM [20] also uses a proactive approach, but focuses on MANETs where
nodes decide to communicate with one another based on their location.

SDAR [12], PAR [28], CAR [92] and ANODR [58] use a reactive approach, in which a
sender broadcasts a route request that is flooded throughout the network, until it reaches
the intended recipient. All nodes that receive the request perform a public key operation,
in order to determine whether they are the intended recipient or not. The schemes dif-
fer in how they modify the broadcasts so as to establish an anonymous channel between
sender and receiver and by how communication proceeds once a path has been established.
AnonDSR [93] and ARM [91] also use an on-demand approach to route discovery alongside
onioning and public key cryptography, but attempt to improve upon the security, anonymity
and performance of SDAR, ANODR and MASK by using pseudonyms, dummy traffic and
padding. Acimin [34] is an on-demand DC-net based protocol that uses layered encryption
during routing. ODAR [94] is an on-demand source routing based protocol that uses Bloom
filters [9] to conceal node identities. Instead of appearing in the clear, “true” identities are
cryptographically hashed and placed in Bloom filters, such that a node can silently check
whether it is part of a path by testing for membership in the filter.

To the best of our knowledge, existing approaches to anonymity in MANETs attempt
to deal with a global eavesdropper that can perfectly track nodes and messages, or that
can identify nodes by revealing their logical identities. In contrast, our approach deals
with a much weaker adversary and acknowledges that logical identities are no more than a
mechanism for enabling routing. Some attempt to create an “anonymous route”, in which
the “true” identity of a route member is known only to a limited set of nodes, such as to
the sender or to the member’s neighbors in the route. Throughout this work, we attempt
to decouple anonymity from routing and note that a node has no “true” identity.

\textsuperscript{1}When considering a system with nodes running multiple algorithms, a node can be identified by the
type of algorithm it runs. By piggybacking on existing algorithms, we achieve “type anonymity” [47].
Chapter 11

Conclusion

In this chapter, we summarize and conclude our work. In section 11.1, we give a brief summary of previous chapters. In section 11.2, we discuss our results and their implications. In section 11.3, we summarize our contributions. In section 11.4, we offer gleaned insights concerning anonymity in MANETs. Lastly, in section 11.5, we set up future work.

11.1 Summary

In this work, we briefly investigated anonymous routing in mobile ad hoc networks. At first glance, the requirements of anonymity and routing appear to conflict, in that symmetry must be broken yet maintained. But, upon deeper inspection, the algorithms investigated in this work reveal the contrary. On the one hand, one can avoid asymmetry by design. On the other, one can compensate for existing asymmetry with the aid of cryptography.

In chapter 2, we presented several models to support our investigation. We defined mobile ad hoc networks and their emissions, explored the threats posed to their anonymity by attackers of different strengths and derived suitable adversary and trust models.

In chapter 3, we presented the goals of our design and stated the problem to be solved. We defined anonymity, related concepts and an information hiding order on distributed systems. We defined the notion of a point to point communication system and gave a template for routing algorithm procedures. Then, we defined identities, identity agreement and identity-based routing. Lastly, we briefly noted desirable performance traits.

In chapter 4, we used our terminology to specify algorithms that were designed without anonymity in mind. In chapter 5, we set the stage for their analysis. In chapter 6, we proved that an unbiased identity agreement helps achieve observational anonymity. In chapter 7, we proved that a plurality of identities offers an increase in information hiding.

In chapter 8, we presented algorithms that were designed with anonymity in mind. In chapter 9, we compared all of the algorithms presented in this work, in terms of their attitude towards bias. Finally, in chapter 10, we described additional related work.

We note again that our investigation focused on the effect of network layer addresses on anonymity, against an arguably stunted adversary. Our results are accordingly confined. As mentioned, the reason for choosing to ignore aspects such as spacetime, message contents and identity agreement implementation was to curtail the complexity of our analysis.
11.2 Discussion

Our definition of anonymity portrays it as a product of two factors – symmetry among the nodes and adversary strength. On the left hand side, the symmetry in question is accounted for by an isomorphism relation, $\simeq$. On the right hand side, the adversary in question is accounted for by an indistinguishability relation, $\simeq$. Both are conveniently substitutable.

With the goal of anonymity in mind, we considered a weak adversary and showed that anonymity can be achieved by use of unbiased identity agreements – though identities must be unique, they must not be uniquely given. The strength of our results stems not from their applicability to a wide array of scenarios, but rather from their demonstration of the viability of our definition – they suggest that striving for symmetry aids anonymity.

In our approach, addresses are not strongly associated with destinations. Hence, other criteria must be taken into account when a sender chooses which value to pass to `Network-Layer-Send`. Albeit confusing (on account of the blurred distinction between private and public destination representations), to be accurate, it is these criteria that ought to be associated with “who” the message is destined to, rather than a destination address. Put differently, assuming that input is derived in a non-random manner, it likely codes the semantics of communication – the *why* a node wishes to copy information to another. In a sense, this is the only true way in which to identify the receiver. In our approach, addresses cannot be assigned meaning (whether this poses a drawback must be studied further).

On a related note, a receiver that is unable to verify “who” the sender is cannot tell apart a “real” sender from a “fake” one. In this case, a man-in-the-middle attack is possible, where a “fake” sender relays the information back and forth from a “real” sender\(^1\).

Such identity theft is to be expected, because addresses are not linked to stronger identifiers to begin with. As our approach deals with anonymity only, and does not make use of non-forgeable identities (such as those afforded by public key cryptography), it is indeed vulnerable to this attack. However, the attack highlights an interesting feature of polyonymous routing. In MANETs with forgeable identities, nothing prevents an attacker from assuming the identity of some node, that is perhaps in a different location or that has left the MANET. If the identity belongs to a node of importance, that many nodes may contact for whatever reason, then the attacker is presumably better off. But, we find that in preventing the attacker from narrowing candidates for attack, polyonymous routing also prevents him from knowing “who” to impersonate (i.e., what address to assume).

In order to prevent identity theft, some form of agreement is required, for the sake of bootstrapping authentication. To see this, consider the following two disjoint cases:

1. In the case of symmetric (e.g., HMAC) verification, in order to verify a signature, a receiving verifier must use the correct symmetric key, which has been pre-agreed upon with the sender and used for computing the signature. Hence, it must either already

\(^1\)As an anecdote and an example of this sort of identity theft outside the context of MANETs, consider an acquaintance of the author’s, who would frequent online chat rooms, in which participants start personal chats with random participants, not caring who the random participant they are chatting with is nor requiring strong credentials. This acquaintance would wait for a participant to initiate a personal chat and then relay the messages to a second participant, as though they were his own. Any responses would be relayed back as well. In this way, both participants are in contact with the acquaintance, but neither one of them realizes that they are really reading the messages of someone other than this MitM they are in contact with.
have some indication of which sender is in question (in order to know which key to use), or use all of the symmetric keys in its possession for verification, one by one.

In the former case, the receiver’s knowledge of the right key is similar to the receiver’s knowledge and possession of an alias known by the sender. In both, pairwise trust, anchored in the form of pre-agreement, is required. In the latter case, we suffer degraded performance and an internal impostor can impersonate other nodes.

2. In the case of asymmetric (e.g., RSA, DSS) verification, in order to verify a signature, a receiving verifier must use the right asymmetric public key, whose corresponding private key is owned by the sender and used for computing the signature. No indication of the sender is required, as the public key can be sent alongside the signature.

However, verification is meaningless unless the public key is proved to belong to the sender. If ownership cannot be proven, then a man-in-the-middle attacker can replace the sender’s transmitted public key with one of his own, compute a signature of his own and pass verification. Hence arises the need for an agreement and a PKI (with trust based in pre-installed trusted CA certificates, or in a web of trust [102]).

In both cases, the required agreement leads to a striking resemblance between the two procedures for signature generation and verification and the two procedures for identity encoding and decoding. The same applies to procedures of encryption and decryption. The resemblance indicates that, in terms of agreement overhead, our approach is no worse than the ones of chapter 8 – all require similar agreements in order to properly function.

Lastly, although identity agreement implementation is outside the scope of this work, we note that when nodes use their PRNGs to draw their addresses, all PRNGs must perform equally well or equally badly in order to ensure lack of bias. Otherwise, ironically, a node using a good PRNG in the midst of others using a bad PRNG will still stand out.

11.3 Contributions

Our contributions are briefly summarized as follows:

1. We present models with which to investigate anonymity in MANETs.

2. We present an informal exploration of attacker capabilities and modi operandi. Our exploration serves to derive a generally applicable, view-based, adversary model.

3. Based on our adversary model, we formally define information-theoretic anonymity, related concepts and an information hiding order on distributed systems.

4. We present a general, simple, template for routing algorithm specification.

5. We prove that unbiased identity agreement algorithms achieve anonymity, whereas biased ones do not. In so doing, we question the assumption that logical identities reveal “true” identities and offer a new outlook on their role in anonymization.

6. We introduce the notion of polyonymous routing and analyze its effect on information hiding. In so doing, we question the assumption of one identity per participant.

7. We present a brief comparison of our work and that of others.
11.4 Insights

Our insights are briefly summarized as follows:

1. Anonymity is a form of symmetry among processors and depends on the adversary at hand. A logical conclusion is that anonymization is a form of symmetrization.

2. Identifiers are not required to be hidden, only their relationship to their owner is. In anonymization schemes built for the Internet, where an address is linkable to its owner, a node must hide its identity behind the identities of others, while informing as few other nodes as possible of its identity. In this way, it is ensured that nodes’ collective knowledge of the identities of others involved in the process of communication is minimized. In this work, we proposed a qualitatively different strategy for MANETs. A sender informs the network what node it desires to contact by specifying an address that is specifically chosen so as to not reveal which node is being contacted.

3. Any system component may violate anonymity, by causing certain features to stand out. Hence, in contrast with industry-standard security protocols [56][23], anonymity cannot be expected to be implemented at one layer and hold sway over others.

4. Given the economics of anonymity, one may wish to be similar to others, in order to diminish the chance of a successful attack. Case in point, one wishes to use the same PRNG as everyone else, in order to avoid standing out. In contrast, given the economics of security, one may wish to be slightly dissimilar from others, in order to diminish the chance of a successful attack. Case in point, exploits written for a popular operating system are useless against a user who does not use it.

5. What is classically considered an attack can be turned in favor of defense. In the one case, we use multiple identities, à la Sybil attack [27], to prevent an attacker from narrowing down candidates for attack. We note that the prevention of Sybil attacks, as in [21], interferes with our approach. In the other case, the use of wormholes is proposed as a way to achieve network topology randomization (see Appendix C).

11.5 Open issues

Our work leaves the following issues for future work:

1. Analyzing the algorithms proposed in this work under stronger adversaries. A strong adversary is likely able to correlate addresses with nodes that own them. It may be interesting to see under what circumstances and to what degree this is true.

2. Obtaining a natural measure of anonymity, numerical or otherwise. At the time of this writing, the literature suggests a lack of consensus on the matter. We briefly mention a potential candidate, for the sake of future study. Given an algorithm and an adversary, one can observe the complete bipartite graph over inputs and “colors”, in which arcs correspond to events of the form $E_{ALG}^{x,c}$ and are weighted according to their respective probabilities. Since every pair of indistinguishable inputs gives an automorphism of this graph, its automorphism group may yield a good measure.
3. Examining additional unbiased agreements not studied in this work. For instance, as a more granular alternative to per-message agreements, one might consider letting nodes renew their identities only every so often, according to a temporal parameter or according to the number of packets sent. Or, perhaps, nodes might switch identities with one another during runtime, as children do to taunt substitute teachers.

4. Implementing unbiased mononymous and polyonymous agreements in a simulator.

5. Testing the performance of various routing algorithms under different agreements. The agreements presented in this work are not expected to scale well, given that random addresses make route summarization more difficult. In a network that carries many routes, this can have a dire impact on routing table sizes. One might ask whether this performance hit can be mitigated without compromising system anonymity.

6. Implementing identity agreements in an anonymous way, perhaps using symmetric secret key exchange schemes such as [10] and [8], or MANET adaptations of these.

7. Under polyonymous routing, the worst-case space complexity at a receiver increases. We wish to reduce this complexity, or show a lower bound. We note that despite the apparent increase, in practice, a receiver is only required to hold as many addresses as there are senders contacting it simultaneously (or incoming messages arriving simultaneously, in the case of per-message agreements). We wish to determine whether such complexity is overshadowed by buffer allocations of the transport layer.

8. Reformulating our approach using a more general “virtual network” abstraction and enabling nodes to controllably correlate between different “world views”, in order to increase routing efficiency (or, at least, attempt to make correlations explicit).


10. Adapting the notion of unbiased agreements for use with multicast.

11. Analyzing the impact of agreements on motion analysis attacks.

12. Augmenting anonymity with secrecy and integrity guarantees.
Appendix A

Optional features

At a bare minimum, a routing algorithm must ensure correct delivery. However, routing algorithms can achieve more. In this appendix, we briefly explain additional features that may be implemented by a routing algorithm, using the terminology of this work.

Sender indication

Route-On-Send adds an indication of the sender to the message.

Routers and routes

A routing algorithm may implement the notion of a “route”, or a sequence of designated nodes in charge of relaying a message. The mechanism by which route members are chosen may be different per algorithm, but in order to amortize the costly route setup, routes are (almost by definition) expected to be reused for relaying more than one message.

Since route members behave differently from non-members, a computation must break symmetry, such that only members perform a certain action (e.g., forward a message). To this end, a routing algorithm may use router indicators, defined analogously to destination indicators. Although destination and router indicators are similar in concept, they differ in the extent to which knowledge is distributed in the network. Whereas knowledge concerning available destinations must span the entire network in the worst case, knowledge concerning available routers is typically kept more localized, for reasons of efficiency (in terms of communication complexity, global data structures can be more costly to maintain).

As with destination indicators, the use of router indicators must be coordinated by the use of an agreement algorithm. The algorithm may generate router indicators per router, per router per route or per router per route per message. Although router indicators and destination indicators can be chosen from the same space without affecting correctness, we note that use of disjoint spaces makes it harder for an attacker to use observed router indicators to logically access (and potentially compromise) nodes, thereby decreasing the overall attack surface of the system. In particular, use of disjoint spaces for these indicators may also prevent the iterative node geolocation attack described in section 2.4.2.
Router indication

Route-On-Recv adds an indication of the router to the message.

Route indication

Nodes compute a route indication as an aggregate function of router indications, or in an algorithm-specific manner. The indication can be used by the sender, recipient and routers along the way for bookkeeping purposes or for certain types of optimizations.

Source routing

Route-On-Send adds a “desired route” route indication to the message.

Silent reception

If possible, Accept initially invokes Route-On-Recv (or a special substitute), in order to partially simulate a non-receiver. While it is only the intended receiver that delivers the message to the upper layers, the simulation increases the observed similarity among processors.

Simple return traffic

Given an indication of the route, \( i_{\text{route}} \), and an indication of the previous router, \( i_{\text{prev}} \), Route-On-Recv records \( i_{\text{prev}} \) in a data structure, indexed by \( i_{\text{route}} \). A recipient that wishes to reply to an incoming message adds \( i_{\text{route}} \) and a “return traffic” flag to its outgoing response. Then, when Route-On-Recv receives an incoming message with a “return traffic” flag, it routes it back to the sender through the recorded previous router.

Sessions

If the information to be copied between nodes proves to be too numerous or not all available at the same time, it must be sent in pieces. In order to reassemble the pieces, an algorithm may employ the notion of a “session” in order to group messages. While sessions are a responsibility of the upper layers, we discuss them here due to their relation to identities.

In order for a session to occur, an endpoint must be able to link packets to each other. If an incoming message does not contain sender indication, and multiple senders are sending messages to the same receiver, the receiver cannot tell the different senders apart. If this is the case, then the receiver cannot link successive packets as belonging to the same sender, unless other flow identification methods exist (e.g., a unique port number).
Two-way sessions

When a two-way session is required, a communication system may:

1. Explicitly handle return traffic, possibly in the simple manner described above.

2. Rely on sender indication. A sender that wishes to receive return traffic includes a source indicator in the message, such that the receiver can determine which of its own target values correspond to the sender. When the receiver wishes to send return traffic, it chooses this value and passes it to Network-Layer-Send as input.

   We pedantically note that sender indication means the receiver is able to map the source indicator included by the sender to a private representation of a sender, and from there, the private representation of the sender to a corresponding private representation of a destination, for subsequent use with Network-Layer-Send.

3. Rely on a return address mechanism\(^1\). A sender that wishes to receive return traffic includes an identity of its own as part of a “return address” field in the message. When the receiver wishes to send return traffic, this destination indicator is simply copied over to the return message as it is sent out.

In the case of non-explicit handling of return traffic, in order for the sender to be able to link incoming return traffic to previous outgoing traffic, the return traffic must include some form of flow identification (e.g., a port number, a special return address chosen for the session, or use of sender indication by the traffic returner).

\(^1\)The subtle difference from sender indication is that a return address mechanism requires no destination-related state at the receiver. Given the footnote of section 3.2.3, it is unclear whether an algorithm in which a receiver switches source and destination addresses employs a return address mechanism, or rather sender indication under the assumption of identical sender source and destination indicators.
Appendix B

Procedures of DSR

Implementations of Route-On-Send, Filter, For-Me, Accept and Route-On-Recv are shown in Figures B.1 through B.5, respectively. They are explained below:

1. Given a message to send, DSR-Route-On-Send retrieves a stored route to the destination, adds it to the message as the source route field and unicasts the message to the first member of the route. If a stored route is not found, DSR-Route-On-Send calls an auxiliary procedure, Establish-Route, and waits until a route is established before sending the message. Establish-Route issues a flood search, also known as a route request, by invoking an FRA as a black box (by calling FRA-Network-Layer-Send).

2. DSR-Filter filters RREQs by invoking FRA-Filter as a black box, filters RREP's not matching a previously stored RREQ identifier and never filters DATA messages.

3. DSR-For-Me rejects RREQs when FRA-For-Me rejects them, rejects RREP's when they did not originate at the processor to begin with and rejects DATA messages when the processor is not listed as the destination.

4. When DSR-Accept accepts an RREQ, it issues an RREP in response. When DSR-Accept accepts an RREP, it signals the completion of route establishment and stores the learned route for later use. When DSR-Accept accepts a DATA message, it delivers the message to the upper layers.

5. If an RREQ is received, DSR-Route-On-Recv adds the processor’s address to the route field and invokes the FRA as a black box. If an RREP is received, the RREP is broadcast to all the neighbors. If a DATA message is received, DSR-Route-On-Recv strips the processor’s address from the source route field and unicasts the message to the next router listed in the source route field.

Destination representations are handled in DSR-Encode-Destination-Indicator and DSR-Decode-Destination-Indicator, shown in Figures B.6 and B.7, respectively. In addition, DSR makes use of router indicators, which are handled by DSR-Encode-Router-Indicator and DSR-Decode-Router-Indicator, shown in Figures B.8 and B.9, respectively. Together, these four procedures encapsulate the handling of network layer identities in the algorithm.
Procedure DSR-Route-On-Send($m$: message, $t$: target)

Initially, $RouteDB = \phi$, $RREQDB = \phi$

begin
  if $RouteDB[t] = \bot$ then
    call Establish-Route($t$)
  wait until $RouteDB[t] \neq \bot$

  $M := \text{new message}$
  $M.type := \text{DATA}$
  $M.dest := \text{Encode-Destination-Indicator}(t)$
  $M.sroute := RouteDB[t]$ /* set source route field */
  $r := \text{Decode-Router-Indicator}(M.sroute[0])$
  call Datalink-Unicast($M$, $r$) /* hand message to 1st router */
end

Procedure Establish-Route($t$: target)

begin
  $M := \text{new message}$
  $M.type := \text{RREQ}$
  $M.route := \text{Encode-Router-Indicator}(me_i)$
  $M.rreqid := \text{Generate-RREQ-Identifier}(t)$
  $RREQDB[M.rreqid] := t$
  call FRA-Network-Layer-Send($M$, $t$)
end

Figure B.1: Route-On-Send() Procedure for DSR

Procedure DSR-Filter($m$: message)

begin
  switch $m.type$:
    case $\text{RREQ}$:
      /* black box criteria */
      if FRA-Filter($m$) = true then
        return true
    case $\text{RREP}$:
      /* own criteria */
      if $RREQDB[m.rrepid] = \bot$ then
        return true
    case $\text{DATA}$:
      /* own criteria */
      $r := \text{Decode-Router-Indicator}(M.sroute[0])$
      if $r \neq me_i$ then
        return true
    return false
end

Figure B.2: Filter() Procedure for DSR
Procedure DSR-For-\(m\): message) 
begin 
switch \(m\): type: 
\(\text{case } \text{RREQ:} opposes */
   \text{if } \text{FRA-For-}\(m\) = false then
      \text{return false}
\(\text{case } \text{RREP:} /* own criteria */
   r := \text{Decode-Router-Indicator}(m\):route[0])
   \text{if } r \neq \text{me}_i \text{then}
      \text{return false}
\(\text{case } \text{DATA:} /* own criteria */
   t := \text{Decode-Destination-Indicator}(m\):dest)
   \text{if } t \neq \text{me}_i \text{then}
      \text{return false}
\text{return true}
end

Figure B.3: For-\(m\): Procedure for DSR

Procedure DSR-Accept\(m\): message) 
begin 
switch \(m\): type: 
\(\text{case } \text{RREQ:}\
   M := \text{new message}
   M\):type := RREP
   M\):route := m\):route
   M\):rrepid := m\):rreqid
   \text{call Datalink-Broadcast}(M)
\(\text{case } \text{RREP:} /* retrieve destination, indexed by RREP identifier */
   t := \text{RREQDB}[m\):rrepid]
   RouteDB[t] := m\):route /* store route, release waiting DSR-Route-On-Send */
\(\text{case } \text{DATA:} /* retrieve destination */
   \text{deliver } m\):msg
end

Figure B.4: Accept\(m\): Procedure for DSR
Procedure DSR-Route-On-Recv(m: message)
begin
switch m.type:
case RREQ:
    m.route := m.route + Encode-Router-Indicator(me)
    t := Decode-Destination-Indicator(m.dest)
    RREQDB[m.rreqid] := t /* store destination, indexed by RREQ identifier */
    call FRA-Route-On-Recv(m) /* continue broadcasting RREQ if necessary */
case RREP:
    call Datalink-Broadcast(m)
case DATA:
    m.sroute := m.sroute - m.sroute[0]
    r := Decode-Router-Indicator(m.sroute[0])
    call Datalink-Unicast(m, r)
end

Figure B.5: Route-On-Recv() Procedure for DSR

Procedure DSR-Encode-Destination-Indicator(i: target)
begin
    return FRA-Encode-Destination-Indicator(i)
end

Figure B.6: Encode-Destination-Indicator() Procedure for DSR

Procedure DSR-Decode-Destination-Indicator(a: address)
Initially, DestinationDB[a] = ⊥ for all a
begin
    if DestinationDB[a] ≠ ⊥ then
        return DestinationDB[a]
    t := FRA-Decode-Destination-Indicator(a)
    if t = ⊥ then
        /* private representation of a is unavailable */
        t := new target
        /* but needed by caller, so make new one */
        DestinationDB[a] := t
        /* and store it in case a is seen again */
    return t
end

Figure B.7: Decode-Destination-Indicator() Procedure for DSR
Procedure DSR-Encode-Router-Indicator\(_i\)\((r: \text{router})\)

\(RIDs_i\) is the set of router indicators used by processor \(p_i\)

Denote the sole router indicator used by \(p_i\) with \(const_i\)

\begin{align*}
\text{begin} \\
\quad \text{if } r = me_i \text{ then} \\
\quad \quad \text{return } const_i \\
\quad \text{else} \\
\quad \quad \text{return } \perp \\
\text{end} \\
\end{align*}

/* public representation of \(r\) is unavailable */

Figure B.8: Encode-Router-Indicator() Procedure for DSR

Procedure DSR-Decode-Router-Indicator\(_i\)\((a: \text{router indicator})\)

\(RIDs_i\) is the set of router indicators used by processor \(p_i\)

\begin{align*}
\text{begin} \\
\quad \text{if } a \in RIDs_i \text{ then} \\
\quad \quad \text{return } me_i \\
\quad \text{else} \\
\quad \quad \text{return } \perp \\
\text{end} \\
\end{align*}

/* optional use of Bloom filter [9] to save space */

/* private representation of \(a\) is unavailable */

Figure B.9: Decode-Router-Indicator() Procedure for DSR
Comments about the code

The code is expressed as it is in order to emphasize that

- The phases of DSR are reflected in the structure of its procedures.
- By our convention, a message is intended for a processor unless explicitly determined otherwise. Similarly, a message is not filtered unless explicitly stated otherwise. This is coded in the structure of DSR-For-Me and DSR-Filter, respectively.
- A designer can choose between invoking another routing algorithm as a black box or explicitly specifying routing logic. In the example of DSR, our code relegates RREQs to a black box FRA. However, even though RREPs can also be relegated to an FRA (one that has a more stringent filter), they are handled explicitly.
- A routing algorithm that uses another routing algorithm as a black box must do so carefully. In particular, it is important for it to filter messages that would have been filtered by the black box and reject messages that would have been rejected by the black box (as opposed to accepting them).
Appendix C

Wormhole routing

We propose *wormhole routing*, or WHR, in which nodes send messages to their final destinations via intermediaries. The set of potential intermediaries forms an overlay, and a random walk on the overlay determines the ones actually used to relay a given message.

In a sense, WHR offers a natural generalization of the forwarding mechanism in Crowds to arbitrary overlays. But despite its similarity to Crowds, the purpose of wormhole routing is not to prevent a server from realizing the address of a client, but rather to scramble the route taken by a message, in order to deny topological knowledge to a local adversary.

The effect of using randomly chosen intermediaries is that a message visits multiple neighborhoods of the datalink layer topology graph before arriving at its final destination, instead of perhaps taking the shortest route possible. In this way, a local eavesdropper cannot directly use observed messages to relate his location to the locations of neighborhoods outside his vicinity, since messages of all neighborhoods pass through his vicinity at random (we assume that messages and addresses do not reveal how neighborhoods are related). Clearly, such an approach offers a tradeoff between security and communication complexity. We conjecture that it masks any node-dependent biases in the datalink layer topology.

C.1 Assumptions

**Network**

- Same as for FRAs, except that WHR is not necessarily available on all nodes. Instead, we assume the existence of an underlying routing algorithm, $ALG$, that is installed on all nodes, and which WHR uses for routing between WHR-aware nodes.

**Adversary**

- We assume a local eavesdropper with perfect spatial eavesdropping and device fingerprinting resolution, whose view contains the messages sent and received by nodes in his vicinity. Given this, the adversary can attribute any message observed in his vicinity to a transmitting device, but cannot directly determine the location of transmitting nodes outside his vicinity or their relationship to nodes within his vicinity.
Trust

- Same as for FRAs. In addition, we assume an overlay in the form of a wormhole graph, $G_W = (V, E)$, that is maintained in a distributed manner. $V$ is the set of WHR-aware nodes and $E$ is the set of logical "wormhole" connections between them. A WHR-aware node maintains a list of neighbors in this graph, each of which can serve as an intermediary when sending a message across the network.

C.2 Specification

Implementations of Route-On-Send, Filter, For-Me, Accept and Route-On-Recv are shown in Figures C.1 through C.5, respectively. They are explained below:

1. Given a message to send, WHR-Route-On-Send uses $ALG$ to either send the message directly to its destination, or to forward it to an intermediary node, according to a flip of a coin. In the former case, the message is marked with a teleport flag set to false and sent as is. In the latter case, the message is marked with a teleport flag set to true and an indication of the final destination. The intermediary node is chosen at random from the processor’s neighbors in the wormhole graph.

2. WHR-Route-On-Recv routes an incoming message as $ALG$ would.

3. WHR-Filter filters a message when $ALG$ does.

4. WHR-For-Me rejects a message when $ALG$ does.

5. If the processor accepting the message is an intermediary, WHR-Accept recursively calls WHR-Network-Layer-Send, in order to send the message to its final destination. Otherwise, it accepts the incoming message as $ALG$ would.

Comments about the code

1. Since we allow $ALG$ and $WHR$ to operate in the network simultaneously, WHR-aware nodes could conceivably receive messages that originate from both WHR-aware and WHR-unaware nodes. In order to behave correctly, WHR-aware nodes must be able to differentiate between the messages. For ease of presentation, we assume that message fields used by WHR are unique to WHR and are evaluated to false (or equivalent) in messages that do not contain them (i.e., originate from WHR-unaware nodes).

2. As it is presented here, WHR is intended for unidirectional routing. Without further modification, an underlying routing algorithm that relies on sender indication for the purpose of bidirectional routing will fail, because a receiver receives a sender indication that is issued by the last intermediary rather than by the original sender.

There are two ways of dealing with this. The first is for WHR to explicitly handle return traffic by maintaining records pointing to the intermediaries from which a message was received. The second is to transfer enough state to the final intermediary, such that it can fully simulate (i.e., impersonate) the sender.
Procedure WHR-Route-On-Send(m: message, t: target)
begin
    $M := \text{new message}$
    $M.msg := m$
    $M.teleport := false$
    if $\text{WHRNeighbors} \neq \phi$ then
        $M.teleport \leftarrow_R \{\text{true, false}\}$
    if $M.teleport = \text{true}$ then
        $M.finaldest := \text{Encode-Destination-Indicator}(t)$
        $T \leftarrow_R \text{WHRNeighbors}$
    else
        $T := t$
    call ALG-Route-On-Send($M, T$)
end

Figure C.1: Route-On-Send() Procedure for WHR

Procedure WHR-Filter(m: message)
begin
    return ALG-Filter(m)
end

Figure C.2: Filter() Procedure for WHR

Procedure WHR-For-Me(m: message)
begin
    return ALG-For-Me(m)
end

Figure C.3: For-Me() Procedure for WHR
Procedure WHR-Accept(m: message)
begin
    if m.teleport = true then
        t := Decode-Destination-Indicator(m.finaldest)
        if t ≠ me, then
            call WHR-Network-Layer-Send(m.msg, t) /* processor is an intermediary */
        else call ALG-Accept(m) /* processor is final destination of teleported message */
    else call ALG-Accept(m) /* processor is regular destination of unteleported message */
end

Figure C.4: Accept() Procedure for WHR

Procedure WHR-Route-On-Recv(m: message)
begin
    call ALG-Route-On-Recv(m)
end

Figure C.5: Route-On-Recv() Procedure for WHR
References


anonymous routing protocol for wireless and mobile ad hoc networks. In LCN
'04: Proceedings of the 29th Annual IEEE International Conference on Local
Computer Networks, pages 618–624, Washington, DC, USA, 2004. IEEE Com-
puter Society.


[15] D. Chaum. Untraceable electronic mail, return addresses, and digital

[16] D. Chaum. The dining cryptographers problem: unconditional sender and re-

[17] S. Cheshire, B. Aboba, and E. Guttman. Dynamic Configuration of IPv4 Link-
Local Addresses. RFC 3927 (Proposed Standard), May 2005.

us/how-to-bluesniper-pt1,review-408.html, March 2005.

In P2P ’06: Proceedings of the Sixth IEEE International Conference on Peer-
Society.


wireless sensor networks. In WOWMOM ’06: Proceedings of the 2006 Interna-
tional Symposium on on World of Wireless, Mobile and Multimedia Networks,

In Dingledine and Syverson [25], pages 54–68.


onion router. In In Proceedings of the 13th USENIX Security Symposium, pages

Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-


[53] L. Joncheray. A simple active attack against TCP.


ניהホームページ ברשויות אודו ניידות

 Nir Roland
ניתוב אנונימי בﴩרותroat חק נידוח

חובר על מחקר

לשם מחיצים חלקיים של הדרישה ל嘉年ה חנור
מניטר למדעי מדעי המחשב

 Nir Rogel

רוחש לכסת הכותבים - מון טכנולוגי לישראל
אפריל 2011.tifafa

ינס תשיעי
המחקטר טעינה במחשיט רעי פיתוח מתכונת בקנולא במדעי המחשב

והותי نوفمبر כלכל על כל. להמסת האורכה על תופרות החובות. להדריר האמצעיים על כל
كرיאת הידוד. להזדהי על סבלנות החובות וקנולא המסמרת שלמותי לאזרחי הזדד.
אמור. רוחי נר וניאו מי חותם רחב, כל עין בברך,-encoded על שיבוש של דר וולך岛上.
דיגאל. על הים ומלוב, על הסולט ארדות המאשיט. ארדות בין לע פינס טווס. כנרות, על
הنموذج, כלה שsockopt"verige שת,proto, delimited. לכל שפילק עמי את עתים האלביה בנסותי רבעי גהוז
万户スタイル והשלון בתי חוץ. ארז, על בתויתות השקטות על השפותי אתihad聯 שטחני
רוחי על השไหนים אומרים על דJvm טו הקושט. על זיליות ה台灣, ועל שחקשים כלכליים
איספונות, על שיאורה עללתני אנדטי, ועל שיאורה עבודות על אזור, על שקרא טריונה איגוספונים.
על טווחי החופיווהון על שזרויה רוחף עם חיקח אידים עד שטוש (יתל תמיכה כוספונים)
ואנ phườngו לשכונת על תופרות הקספיאו הרידריה בשיתוף

Technion - Computer Science Department - M.Sc. Thesis  MSC-2011-06 - 2011
The research was conducted under the guidance of Prof. A. B., and with the collaboration of other researchers. The aim of the research was to develop a new algorithm for solving the problem of...
כותרת:

ברקע הקונסטרוקציה של מערכות",

לפי הוראה, על איה שיא מועצה מיוחדת ברורה.

בעזרת החבילה המינימית ב mạchאֶה

ברקע הקונסטרוקציה של מערכות

לפי הוראה, על איה שיא מועצה מיוחדת ברורה.

בעזרת החבילה המינימית ב mạchאֶה

ברקע הקונסטרוקציה של מערכות

לפי הוראה, על איה שיא מועצה מיוחדת ברורה.

בעזרת החבילה המינימית ב mạchאֶה

ברקע הקונסטרוקציה של מערכות

לפי הוראה, על איה שיא מועצה מיוחדת ברורה.

בעזרת החבילה המינימית ב mạchאֶה

ברקע הקונסטרוקציה של מערכות

לפי הוראה, על איה שיא מועצה מיוחדת ברורה.

בעזרת החבילה המינימית ב mạchאֶה

ברקע הקונסטרוקציה של מערכות

לפי הוראה, על איה שיא מועצה מיוחדת ברורה.

בעזרת החבילה המינימית ב mạchאֶה

ברקע הקונסטרוקציה של מערכות

לפי הוראה, על איה שיא מועцаָה מיוחדת ברורה.

בעזרת החבילה המינימית ב mạchאֶה

ברקע הקונסטרוקציה של מערכות

לפי הוראה, על איה שיא מועצת מיוחדת ברורה.

בעזרת החבילה המינימית ב马克אֶה