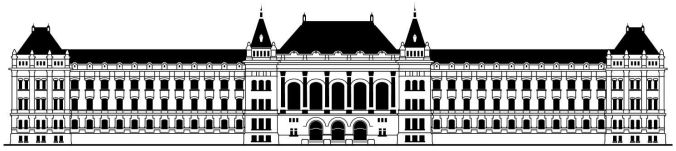


Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics

Department of Telecommunications
Laboratory of Cryptography and Systems Security (CrySyS)

Secure Routing in Multi-hop Wireless Networks

Ph.D. Dissertation
of

Gergely Ács

Research Supervisor:

Levente Buttyán, Ph.D.

2009

Alulírott *Ács Gergely* kijelentem, hogy ezt a doktori értekezést magam készítettem, és abban csak a megadott forrásokat használtam fel. Minden olyan részt, amelyet szó szerint, vagy azonos tartalomban, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

I, the undersigned *Gergely Ács* hereby declare that this Ph.D. dissertation was made by myself, and I only used the sources given at the end. Every part that was quoted word-for-word, or was taken over with the same content, I noted explicitly by giving the reference of the source.

Budapest, 2009. április 8.

. . . . . .

Ács Gergely

Abstract

Routing is a fundamental networking function in every communication system, and multi-hop wireless networks are no exceptions. Attacking the routing service, an adversary can easily paralyse the operation of an entire network. Compared to traditional wired networks, such attacks can be performed relatively easily in wireless networks due to the unsupervised access to the wireless medium. The malicious manipulation of some routing messages results in the dissemination of incorrect routing information which can eventually lead to network malfunction. Even more, intermediate nodes can be corrupted, and thus, exhibit arbitrary behaviour. Considering these facts, securing routing protocols is a primary task, however, designing such secure routing protocols is not a straightforward procedure. A widely used method has been so far to identify different types of possible attacks against routing, and to define routing security implicitly as resistance to these attacks. However, this approach does not provide a common ground for comparing routing protocols in terms of security. Moreover, due to the subtle nature of attacks against routing protocols, such informal reasoning is an error-prone method. In this dissertation, I develop a formal framework in which precise definitions of secure routing can be given, and secure routing protocols proposed for multi-hop wireless networks can be rigorously analysed. I demonstrate the usefulness of this framework in several ways: first, I prove the security of several existing routing protocols that were proposed earlier independently from my work. I show that my model is capable of distinguishing between routing protocols in terms of security. Second, applying the design principles that were identified during the analyses, I propose novel routing protocols for wireless ad hoc and sensor networks and I prove that they are secure in my model.

Kivonat

Az útvonalválasztás alapvető hálózatrétegbeli szolgáltatás minden kommunikációs hálózatban, és ez alól a többugrásos vezetéknélküli hálózatok sem kivételek. Egy támadó az útvonalválasztás megtámadása által az egész hálózat működését könnyen megbéníthatja. Összehasonlítva a hagyományos vezetékes hálózatokkal ezek a támadások relatíve könnyen megvalósíthatóak, mivel a vezetéknélküli közeghez bárki könnyedén, felügyelet nélkül hozzáférhet. Néhány útvonalválasztó üzenet rosszindulatú manipulációja inkorrekt útvonalválasztási információ elterjedését eredményezheti a hálózatban, amely végül a hálózat hibás működéséhez vezethet. Ráadásul a közbenső csomópontok akár korruptak is lehetnek, és így tetszőleges viselkedést mutathatnak. Figyelembe véve ezeket a tényeket, az útvonalválasztás biztonságossá tétele elsődleges feladat, viszont ilyen biztonságos protokoll tervezése nem egyszerű. Egy széleskörben használt módszer a különböző típusú támadások azonosítása az útvonalválasztás ellen, és az útvonalválasztás biztonságának mint a protokoll ellenállóképességének implicit definiálása ezen támadások ellen. Ugyanakkor ez a fajta megközelítés nem biztosítja a protokollok összehasonlíthatóságát biztonság szempontjából. Ezen felül az ilyen informális érvelés hibákat rejthet magában az útvonalválasztás elleni támadások szövevényes volta miatt. Ebben a disszertációban egy olyan formális keretrendszert javaslok, amelyben a biztonságos útvonalválasztás precíz definíciója megadható, és a többugrásos vezetéknélküli halózatokra javasolt biztonságos útvonalválasztó protokollok formálisan elemezhetőek. Ezen keretrendszer használhatóságát többféleképpen demonstrálom: először bebizonyítom több már létező útvonalválasztó protokoll biztonságát, amelyeket munkámtól függetlenül korábban javasoltak. Megmutatom, hogy a modellem képes különbséget tenni az útvonalválasztó protokollok között biztonság szempontjából. Másodszor, alkalmazva a protokollok analízise során azonosított tervezési elveket, új útvonalválasztó protokollokat javaslok vezetéknélküli ad hoc és szenzorhálózatokra, és bebizonyítom ezek biztonságát a modellemben.

Acknowledgements

I would like to thank *Levente Buttyán* for his supervision of my research, and his support and encouragement that helped me to complete my Ph.D. and write this thesis. He is not only a great engineer who was my advisor and colleague, and from whom I learnt a lot during the last five years, but I have also got a friend in him.

I am thankful to the reviewers of my thesis, *Professor Gene Tsudik* and *Dr. Rolland Vida* for the fast and careful reviews, and the large number of useful comments. I am grateful that they reviewed and evaluated my thesis in spite of the limited time that they had.

I want to thank *István Vajda* with whom I worked together on different papers for their valuable suggestions that improved the quality of my thesis, and for the interesting discussions on cryptography.

I am also grateful to the members of the CrySyS Lab., *Boldizsár Bencsáth*, *László Czap*, *László Dóra*, *Tamás Holczer*, *Péter Schaffer*, and *Vinh Thong Ta*, for the illuminating discussions on different technical problems that I encountered during my research. They also provided a pleasant atmosphere which was a pleasure to work in.

I would also like to mention that my research was supported by the High-Speed Networks Laboratory, and the European Commission through the UbiSec&Sens research project.

Last but not least, I want to thank my parents, *Jenő* and *Ilona*, as well as *Mónika*, for their support, endless patience, and love...

Contents

List of Figures

Chapter 1

Introduction and Overview

The rapid spread of wireless networks is mainly caused by their low deployment cost and the more flexible network access that they offer compared to traditional wired networks. These advantages paved the way for new applications where wired networks would fail or would be impractical due to the high deployment costs. Depending on node capabilities and the required networking infrastructure, these applications resulted in different network architectures like wireless ad hoc and sensor networks.

A wireless ad hoc network is a collection of wireless nodes that form a network without any centralized infrastructure (i.e., there is no access points or base stations). The nodes constituting these networks have similar computational and communicational capabilities (like laptops, or Personal Digital Assistants (PDA)) and they frequently show mobility giving a temporary nature to the network. The incentive of ad hoc networks is that in situations in which there is a lack of communication infrastructure or the existing infrastructure is too costly or inconvenient to employ, wireless mobile users may still be able to communicate by the formation of an ad hoc network. There are many foreseen applications of these networks such as students using laptop computers in a campus to participate in interactive lectures, business associates sharing information during a meeting, soldiers relaying information for situational awareness on the battlefield, or emergency disaster relief personnel coordinating efforts after a natural disaster [Broch *et al.*, 1998].

Wireless sensor networks are large scale networks consisting of a large number of tiny sensor nodes and a few base stations. Sensor nodes are spatially distributed autonomous devices using sensors to monitor environmental conditions, such as temperature, sound, pressure, motion or pollutants in a cooperative manner. Although wireless sensor networks were originally motivated by military applications such as battlefield surveillance, there are many foreseen civilian applications like environment monitoring, healthcare applications, home automation, or traffic control. Each sensor node typically consists of a radio transceiver, a microcontroller, a constraint energy source (like a small battery), and one or more sensor devices. As there can be several thousands of nodes in a network, a single sensor node should cost as low as possible.

Both in wireless ad hoc and sensor networks, nodes employ wireless multi-hop communication to convey information through the network. Indeed, as radio interference renders

the direct transmission of information to the remote receiver impractical and inefficient, all nodes perform message transmissions allowing communication among remote nodes that are outside each other's transmission range. This multi-hop communication is even more crucial if we take into account that wireless sensor networks are composed of resource constrained nodes with limited energy supply. Multi-hop communication asks for a routing algorithm to calculate which nodes should forward a particular message in order to deliver that to the remote receiver (destination). Therefore, routing protocols play a fundamental role in wireless network communications. Different network models and applications resulted in a multitude of routing protocols for multi-hop wireless networks in the recent past.

Routing protocols have two main functions in wireless networks. The first is to discover routes between the source and the destination node(s), while the second one is to forward data messages on the discovered routes. As message forwarding on discovered paths is usually a straightforward procedure, I consider only the route discovery part of wireless routing protocols in this dissertation. However, note that some routing protocols, such as location-based routing protocols, are mainly concerned with the second function of routing, and the discovery function is reduced to neighbor discovery instead of route discovery. These protocols are beyond the scope of this dissertation.

Many applications require multi-hop wireless networks to operate correctly even in hostile environments. Security thus becomes a critical issue in these networks. However, some multi-hop routing protocols have not been designed with security requirements in mind. This means that they can badly fail in hostile environments. The severity of routing security is critically high due to at least two reasons. First, subverting the routing service an adversary can easily paralyse the operation of the whole network. For instance, imagine a vehicular application scenario, where sensors deployed along roadside monitor air temperature to inform drivers of the road condition. A misrouted measurement which never reaches the driver's car or it does but too late can lead to serious accidents. Even more, a casual adversary who though does not prevent packets from being delivered but forces the usage of suboptimal routes in terms of energy consumption can cause energy constraint nodes (like sensor nodes) easily to become out-of-order. Second, while in traditional networks the adversary may be physically restricted in accessing wired links, in wireless networks it can manipulate other nodes' communication relatively effortlessly due to the easy access to the wireless medium. The injection of a few forged routing messages or the modification of some existing ones can have devastating effects on the routing performance. In this dissertation, I focus on the routing security of wireless ad hoc and sensor networks. More specifically, I am concerned with the security of the route discovery function of ad hoc and sensor network routing protocols.

1.1 Motivation

Several "secure" routing protocols have been proposed for ad hoc networks (see [Hu and Perrig, 2004] for a survey). However, the security of those protocols has been analysed either by informal means only, or with formal methods that have never been intended for the analysis of this kind of protocols. Although there are some secure sensor network routing protocols in

the literature (such as [Wood *et al.*, 2006; Deng *et al.*, 2002; Perrig *et al.*, 2002; Yang *et al.*, 2006]) these are only applicable to specific sensor applications. Moreover, their security has been analysed only by informal reasoning too, which is an error-prone method. Paradoxically, research on wireless sensor networks has been mainly fuelled by their potential applications in military settings where the environment is hostile. The natural question that may arise is why then security of routing protocols for sensor networks has fallen beyond the scope of research so far. I believe that one important reason for this situation is that the design principles of secure routing protocols for wireless sensor networks are poorly understood today. First of all, there is no clear definition of what secure routing should mean in this context. Instead, the usual approach (e.g., exemplified in [Hu *et al.*, 2002] or [Karlof and Wagner, 2003]), is to list different types of possible attacks against routing in these networks, and to define routing security implicitly as resistance to (some of) these attacks. However, there are several problems with this approach. For instance, a given protocol may resist to a different set of attacks than another one. How to compare these protocols? Shall we call them both secure routing protocols? Or on what grounds should we declare one protocol more secure than another? Another problem is that it is quite difficult to carry out a rigorous analysis when only a list of potential attack *types* are given. How can we be sure that all possible attacks of a given type have been considered in the analysis? It is not surprising that when having such a vague idea about what to achieve, one cannot develop the necessary design principles. It is possible to come up instead with some countermeasures, similar to the ones described in [Karlof and Wagner, 2003], which are potentially usefuly to thwart some specific types of attacks, but it remains unclear how to put these ingredients together in order to obtain a secure and efficient routing protocol at the end.

In order to remedy this situation, I propose to base the design of secure routing protocols for wireless ad hoc and sensor networks on a formal security model. While the benefit of formal models is not always clear (indeed, in some cases, they tend to be overly complicated compared to what they achieve), I attempt to demonstrate their advantages in the context of ad hoc and sensor network routing protocols. I clearly demonstrate that flaws can be very subtle, and therefore, hard to discover by informal reasoning. In particular, I present new attacks against existing secure routing protocols that motivate a more rigorous approach for making claims about the security of ad hoc and sensor network routing protocols, which is the main theme of this dissertation.

1.2 Related work

There are several proposals for secure ad hoc routing protocols (see [Hu and Perrig, 2004] for a recent overview). However, most of these proposals come with an informal security analysis with all the pitfalls of informal security arguments. In [Karlof and Wagner, 2003], the authors map some adversary capabilities and some feasible attacks against routing in wireless sensor networks, and they define routing security implicitly as resistance to (some of) these attacks. Hence, the security of sensor routing is only defined informally, and the countermeasures are only related to specific attacks. In this way, we even cannot compare the sensor routing

protocols in terms of security. Another problem with this approach is the lack of a formal model, where the security of sensor routing can be described in a precise and rigorous way. While secure messaging and key-exchange protocols are classical and well-studied problems in traditional networks [Bellare *et al.*, 1998; Pfitzman and Waidner, 2001], formal modelling of secure routing in sensor networks has not been considered so far. The adversarial nodes are also classified into the groups of sensor-class and laptop-class nodes in [Karlof and Wagner, 2003], but the capabilities of an adversarial node regarding message manipulations are not discussed. There are also some routing protocols proposed for wireless sensor networks with security in mind [Wood *et al.*, 2006; Deng *et al.*, 2003], however, none of them were analysed by formal reasoning.

Although there are a few exceptions, where some attempts are made to use formal methods for the verification of wireless network routing protocols, I show in the sequel that they either use inappropriate assumptions to prove routing security, or they are not general enough to model the security of different routing protocols.

In [Yang and Baras, 2003], the authors try to reach a goal similar to ours but with a different approach. They propose a formal model for ad hoc routing protocols with the aim of representing insider attacks. Their model is similar to the strand spaces model [Guttman, 2001], which has been developed for the formal verification of key exchange protocols. Routing security is defined in terms of a safety and a liveness property. The liveness property requires that it is possible to discover routes, while the safety property requires that discovered routes do not contain corrupted nodes. In contrast to this, my definition of security in case of source routing allows the protocol to return routes that pass through corrupted nodes, because it seems to be impossible to guarantee that discovered routes do not contain any corrupted node given that corrupted nodes can behave correctly and follow the routing protocol faithfully. My definition of security in case of source routing corresponds to the informal definitions given in [Papadimitratos and Haas, 2002] and [Hu *et al.*, 2002].

Another approach, presented in [Marshall, 2003], is based on a formal method, called Cryptographic Protocol Analysis Language Evaluation System (CPAL-ES), which uses a weakest precondition logic to reason about security protocols. Unfortunately, the work presented in [Marshall, 2003] is very much centred around the analysis of Secure Routing Protocol (SRP) [Papadimitratos and Haas, 2002], and it is not general enough. For instance, the author defines a security goal that is specific to SRP, but no general definition of routing security is given. In addition, the attack discovered by the author on SRP is not a real attack, because it essentially consists in setting up a wormhole between two non-corrupted nodes, and SRP is not supposed to defend against this. In my opinion, wormhole attacks are to be against the neighbor discovery mechanism and not against routing in ad hoc networks. On the other hand, the advantage of the approaches of [Marshall, 2003] and [Yang and Baras, 2003] is that they can be automated.

I must also mention that in [Papadimitratos and Haas, 2002], SRP has been analyzed by its authors using Burrows-Abadi-Needham logic (BAN) [Burrows and Needham, 1990]. However, BAN logic has never been intended for the analysis of routing protocols. It has been developed for verifying authentication properties, and there is no easy way to represent

the requirements of routing security in it. In addition, BAN logic assumes that the protocol participants are trustworthy [Burrows *et al.*, 1990]. This assumption does not hold in the typical case that we are interested in, namely, when there are corrupted nodes in the network controlled by the adversary that may not follow the routing protocol faithfully. All in all, the BAN analysis of SRP in [Papadimitratos and Haas, 2002] was inappropriate, which is also confirmed by the fact that even an Active-0-1 adversary can successfully attack it (see [Buttyán and Vajda, 2004] for details).

Another set of papers deal with provable security for cryptographic algorithms and protocols (see Parts V and VI of [Mao, 2004] for a survey of the field). However, these papers are not concerned with ad hoc and sensor routing protocols. The papers that are the most closely related to the approach I used are [Bellare *et al.*, 1998], [Shoup, 1999], [Backes and Pfitzmann, 2004], and [Pfitzman and Waidner, 2001]. These papers apply the simulation paradigm for different security problems: [Bellare *et al.*, 1998] and [Shoup, 1999] deal with key exchange protocols, and [Pfitzman and Waidner, 2001] is concerned with security of reactive systems in general, and secure message transmission in particular. To the best of my knowledge, I am the first who applied the notions of provable security in the context of routing protocols for wireless ad hoc networks.

In the standard simulation paradigm, security is defined in terms of indistinguishability between an ideal-world model of the system (where certain attacks are not possible by definition) and the real-world model of the system (where the adversary is not constrained, except that it must run in polynomial time). While the real-world model (or simply the dynamic model in my framework) still describes the real operation of the protocol participants in my model, instead of the ideal-world model I use the so-called security objective function to specify how a protocol that is under investigation should operate ideally. Particularly, at the end of each simulation run, the security objective function is applied to the routing state of all honest nodes to decide whether the protocol works according to the specified security objective. The protocol is secure if this security objective function results in a "non-acceptable" value only with a negligible probability, where the definition of what is acceptable or not is protocol dependant. This function may be different for different types of routing protocols, but the general approach of comparing the output of this function in the dynamic model to a pre-defined "acceptable" value remains the same.

The more detailed differences of my model compared to the models proposed so far for the analysis of cryptographic protocols [Bellare *et al.*, 1998; Shoup, 1999; Pfitzman and Waidner, 2001] are the following:

- My communication model does not abstract away the multi-hop operation of the network. In addition, I model the broadcast nature of radio communications, which allows a node to overhear the transmission of a message that was not intended to him (see Section 2.2.4 later). I also take into account that a radio transmission can usually be received only in a limited range around the sender.

- My adversary model is different from the standard Dolev-Yao [Dolev and Yao, 1981] model. In the Dolev-Yao model, the adversary can control all communications in the

system. By contrast, in wireless ad hoc and sensor networks, the adversary uses wireless devices to attack the systems, and it is more reasonable to assume that the adversary can interfere with communications only within its power range. Thus, in my model, the adversary can hear only those messages that were transmitted by neighboring nodes, and similarly, the transmissions of the adversary are heard only by its neighbors (see Section 2.2.1 later). In addition, in traditional wired networks, the adversary is not able to manipulate the messages, if the communication parties can reach each other directly without adversarial interaction. On the contrary, in wireless networks, the adversary can also manipulate the communication of those nodes that can hear each other without adversarial relaying, and thus, a direct link between two honest nodes does not guarantee message authenticity as it is described in Subsection 2.1.4.

- In my model, it is a hypothetic scheduler (see Section 2.2.4 later), and not the adversary, that schedules the activities of the honest nodes. In addition, this activation is done in rounds. This leads to a sort of synchronous model, where each participant is aware of a global time represented by the current round number. However, this knowledge will never be exploited in my analyses. The advantage is that I can retain the simplicity of a synchronous model, without arriving to conclusions that are valid only in synchronous systems.

- The simulation-based approach requires the definition of an ideal-world model, which focuses on what the system should do, and it is less concerned about how it is done. As a consequence, the ideal-world model usually contains a trusted entity that provides the intended services of the system in a "magical" way. In my model, compared to [Bellare *et al.*, 1998; Shoup, 1999; Pfitzman and Waidner, 2001], I eliminate the ideal-world model itself, and I introduce the notion of security objectives (see Section 2.2.3 later) for at least three reasons. (Here, I note that a similar approach is used in [Backes and Pfitzmann, 2004], where the integrity property was defined in terms of a security objective of the Needham-Schroeder-Lowe Public-Key Protocol.)

  First, the security objective function captures the idea of the ideal-world model in the standard simulation paradigm. However, compared to the standard approach, security objective functions consider the high variety of security objectives of secure routing protocols in multi-hop wireless networks, which allows one to apply the same framework to prove the security of different routing protocols. Hence, the diverse security objectives of wireless routing protocols can be modelled in a uniform and flexible way, and secure routing protocols which have the same security objective become comparable in this model.

  Second, although it is tempting to consider the state stored in the routing tables of the nodes as the output, an adversary can distort that state in unavoidable ways. This means that if I based my definition of security on the indistinguishability of the routing states in my model, then it may happen that no routing protocol would satisfy it. Hence, I define the output of the dynamic model as a suitable function of the routing

state, which hides the unavoidable distortions in the states. For instance, this function could be the average length of the shortest paths between the sensor nodes and the base station; then, even if the routing tables of the nodes would not always be correct in some sense (e.g., they do not forward on the shortest path), they still use short enough paths and thus the protocol would still be secure given that the average length of all paths is non-acceptable (e.g., it is above a certain threshold) only with a negligible probability.

Third, although the ideal-world model can also be transformed to model the different security objectives of ad hoc and sensor network routing protocols, I believe that the usage of security objective functions makes proofs easier to follow without losing the soundness of the proof technique.

In the standard approach, the ideal-world adversary models the *tolerable imperfections* of the system; these are attacks that are unavoidable or very costly to defend against, and hence, they should be tolerated instead of being completely eliminated. On the contrary, in my model the security objective, and eventually, the security objective function incorporates the tolerable imperfections of my model.

1.3 Results

I propose a mathematical framework, which allows us to define the notion of routing security in wireless ad hoc and sensor networks precisely and to prove that a protocol satisfies our definition of security. In contrast to prior formal models, my model takes into account the specifics of wireless networks including the broadcast nature of communication, the insider adversary who typically cannot interact with all honest nodes in the network, and the diverse security objectives of routing protocols.

I show that my model is capable of distinguishing between different routing protocols in terms of security, if we consider *routing state pollution attacks*, where the routing tables of honest nodes become polluted with incorrect routing entries. Particularly, routing state pollution attacks are based on malicious message manipulations aiming to corrupt the routing entries of honest nodes. A routing entry is a representation of a route towards a particular destination node, which can be the list of identifiers of nodes constituting the route, or the identifier of the next-hop along which there should be a route to the destination with a certain cost. The goal of these attacks is to cause honest nodes to store such (incorrect) routing entries that are not consistent with the underlying network topology, where the definition of consistency is protocol dependant. Therefore, my model is intended to validate the security of routing protocols against routing state pollution attacks. Although routing state pollution attacks are only a small subset of all possible attacks against wireless routing, I will show that even some of the existing "secure" routing protocols are vulnerable to these attacks.

I list the contributions of my dissertation in more details as follows. I additionally give the corresponding publications and chapters of this dissertation that describe a particular contribution in all details.

- I propose a new and general mathematical framework that consists of a model and

a proof technique, which allows us to define the notion of routing security precisely, to model a given routing protocol, and to prove that a routing protocol satisfies the definition of security in this model.
Related publications: [Ács *et al.*, 2006b; Ács *et al.*, 2006a; Ács and Buttyán, 2007; Ács and Buttyán, 2006; Ács and Buttyán, 2005a; Ács and Buttyán, 2005b; Ács and Buttyán, 2008b]
Chapter: 2

- I designed a novel secure source routing protocol, called endairA, for wireless ad-hoc networks. endairA is the reverse of Ariadne, because, instead of signing the request like in Ariadne [Hu *et al.*, 2002], I propose that intermediate nodes should sign the route reply.
  Related publications: [Ács *et al.*, 2006b; Ács and Buttyán, 2005a; Ács and Buttyán, 2005b]
  Chapter: 3, Section: 3.5.2

- I adapt my security model to dynamic source routing in wireless ad-hoc networks, and I prove that endairA is secure in this model.
  Related publications: [Ács *et al.*, 2006b; Ács and Buttyán, 2005a; Ács and Buttyán, 2005b]
  Chapter: 3, Sections: 3.5.3 and 3.2

- I adapt my security model to dynamic distance vector routing in wireless ad-hoc networks, and I prove that Authenticated Routing for Ad hoc Networks (ARAN) [Sanzgiri *et al.*, 2002] is secure in this model.
  Related publication: [Ács *et al.*, 2005a]
  Chapter: 4, Sections: 4.2 and 4.5.2

- I adapt my security model to link-state routing in wireless sensor networks, and I prove that Intrusion-Tolerant Routing in Wireless Sensor Networks (INSENS) [Deng *et al.*, 2002] is secure in this model.
  Related publication: [Ács *et al.*, 2007]
  Chapter: 5, Sections: 5.2 and 5.4

- I propose a novel secure decentralized label-switching routing protocol called Secure Tiny Lightweight Underlay Adhoc Routing for Wireless Sensor Networks (Secure-TinyLUNAR) that is the secure variant of Tiny Lightweight Underlay Adhoc Routing for Wireless Sensor Networks (TinyLUNAR) [Osipov, 2007] for wireless sensor networks. Secure-TinyLUNAR only uses cost-effective message authentication codes based on symmetric key cryptography.
  Related publication: [Ács and Buttyán, 2008a]
  Chapter: 6, Section: 6.2

- I adapt my security model to label-switching routing in wireless sensor networks, and I prove that Secure-TinyLUNAR is secure in this model.

Related publication: [Ács and Buttyán, 2008a]
Chapter: 6, Sections: 6.4 and 6.6

1.4 Outline

The outline of this dissertation is as follows. In the first part of Chapter 2, I define the context and the scope of this dissertation, which also includes the problem statement. The second part of Chapter 2, along with Chapters 3, 4, 5, and 6 detail the proposed solution and the contributions of my dissertation. The benefit and the current applications of my work are discussed in Chapter 7. Finally, Chapter 8 concludes my work.

Chapter 2: Based on widely used taxonomies, I introduce the problem of routing in wireless ad hoc and sensor networks. I give an overview of some mainstream wireless routing protocols, where I focus on those protocols whose security is analysed later in this dissertation. This chapter also addresses the problem of secure routing in wireless ad hoc and sensor networks. In particular, I specify the adversary model against both ad hoc and sensor network routing. I also list the attack methods used by the adversary to subvert the routing service, and I informally define routing state pollution attacks which are considered in this dissertation. As these attacks can be very subtle, they are difficult to discover. Thus, using informal reasoning exclusively to analyse the security of wireless routing protocols can be dangerous. Hence, I advocate a more systematic and rigorous approach to prove the security of ad hoc and sensor network routing protocols.

In the second part this chapter, I propose a mathematical framework to prove the security of routing protocols in wireless ad hoc and sensor networks. In contrast to prior works, this model takes into account the specifics of wireless networks including the broadcast nature of communication, the insider adversary who typically cannot interact with all honest nodes in the network, and the diverse security objectives of routing protocols. As a first example, I show that the secured version of TinyOS beaconing [Perrig *et al.*, 2002] is insecure in this model considering a minimal security objective of sensor network routing.

Chapter 3: This chapter addresses the problem of secure dynamic source routing in wireless ad hoc networks. I propose a novel source routing protocol, called endairA, where instead of signing the route request messages each intermediate node only signs the reply messages. I also adapt my model to secure source routing and prove that, in contrast to Ariadne [Hu *et al.*, 2002], endairA is secure in that model.

Chapter 4: This chapter discusses the problem of secure dynamic distance vector routing in wireless ad hoc networks. I adapt my model to secure distance vector routing, and I prove that, in contrast to Secure Ad hoc On Demand Distance Vector routing (SAODV) [Zapata and Asokan, 2002], ARAN [Sanzgiri *et al.*, 2002] is secure in that model.

Chapter 5: This chapter addresses the problem of secure centralized link-state routing in wireless sensor networks. I adapt my model to secure centralized link-state routing,

and I prove that INSENS [Deng *et al.*, 2002], which is proposed independently from my work, is secure in that model.

Chapter 6: In this chapter, I deal with the problem of secure label-switching routing in wireless sensor networks. In particular, I propose a novel secure label-switching routing protocol for wireless sensor networks called Secure-TinyLUNAR. Secure-TinyLUNAR is the secure variant of TinyLUNAR [Osipov, 2007] which uses the label-switching routing paradigm to reduce the addressing overhead during data packet forwarding. I adapt my model to secure label-switching routing, and I prove that Secure-TinyLUNAR is secure in that model.

Chapter 2

Modelling Routing Security in Wireless Networks

In this chapter, after describing the context of this dissertation, I propose a mathematical framework, which allows us to define the notion of routing security in wireless ad hoc and sensor networks precisely and to prove that a protocol satisfies our definition of security. The model is general and flexible in the sense that it considers the variety of routing security objectives in multi-hop wireless environments. In contrast to prior formal models, my model takes into account the specifics of wireless networks including the broadcast nature of communication, the insider adversary who typically cannot interact with all honest nodes in the network, and the diverse security objectives of routing protocols.

I show that attacks against wireless routing protocols can be very subtle, and thus, making claims about the security of a routing protocol based on informal arguments only is dangerous. Hence, I propose a novel framework to prove the security of wireless routing protocols. It is important to emphasize that the proposed framework is best suited for proving that a protocol is secure (if it really is), but it is not directly usable to discover attacks against routing protocols that are flawed. Note, however, that such attacks may be discovered indirectly by attempting to prove that the protocol is secure, and examining where the proof fails.

The outline of this chapter is as follows. In Section 2.1, I describe the context and the scope of this dissertation by informally defining the problem of routing and secure routing in wireless ad hoc and sensor networks. Particularly, in Subsection 2.1.1, I shortly introduce wireless routing protocols. I also introduce the operation of some mainstream wireless routing protocols in Subsection 2.1.2, whose security is analysed in this work. Then, in Subsections 2.1.3, 2.1.4, and 2.1.5, I present the objectives of the attacks against routing, and the adversary model including the general attack methods against wireless routing. I also exemplify some attack methods against existing (insecure) wireless routing protocols by illustrative examples. Then, in Section 2.2, I describe a novel framework to prove the security of wireless routing protocols. In particular, in Subsection 2.2.1, I present the adversary model of this framework. Then, in Subsections 2.2.2 and 2.2.3, I describe the network model and the security objective

function, resp. Subsection 2.2.5 specifies the dynamic model and Subsection 2.2.5 introduces the general form of the definition of routing security. In Subsection 2.2.6, I present the proof technique that is used to prove the security of different routing protocols in the rest of this dissertation. As an example for the usage of my framework, in Subsection 2.2.7, I show that the secured Tiny Operating System for wireless embedded sensor networks (TinyOS) beaconing [Perrig *et al.*, 2002] is insecure in my model considering a minimal security objective of sensor network routing protocols. Finally, in Section 2.3, I summarize the chapter.

2.1 Security of Routing in Wireless Networks

In this section, I introduce the problem of routing and secure routing in multi-hop wireless networks and define the context of this dissertation. Specifically, I give a short introduction into wireless routing protocols and I present two classifications of them. I also give a brief overview of the operation of those protocols whose security is considered in this dissertation. Then, I introduce the problem of secure routing in wireless networks. This involves the specification of the adversary model which includes the attack methods against wireless routing protocols.

2.1.1 Classification of Wireless Network Routing Protocols

Routing is a pivotal element of network communications. While, in traditional (wired) networks, the routing functions are performed by special nodes, called *routers*, this does not hold in general for wireless networks. For instance, in wireless ad hoc networks, all nodes perform message transmissions allowing communication among nodes that are outside each other's transmission range. Wireless nodes use a routing protocol to dynamically discover paths, which may traverse several nodes, to any other node. Routing is concerned with ensuring the delivery of messages from a source to some destinations. This involves two functions: (1) the discovery of routes from the source to the destinations, and (2) the forwarding of the messages via the discovered routes. Radio interference, the lossy characteristic of wireless links, and potential node mobility makes routing a challenging task in wireless networks.

Besides ensuring the delivery of messages, routing protocols in most wireless networks have additional objectives. In particular, some protocols are concerned with real-time requirements and aim at minimizing the message delivery time, while others try to maximize the lifetime of the network by minimizing and balancing the energy consumption of the nodes.

The different objectives and application environments of wireless networks resulted in a wide spectrum of wireless network routing protocols (see e.g., [Al-Karaki and Kamal, 2004] or [Liu and Kaiser, 2003] for overviews). These protocols can be classified in many different ways. A simple classification that suits my purposes can be as follows:

- *Topology-based routing protocols*: These protocols typically build a routing topology during the route discovery process that is used later for data forwarding towards the base station. Topology-based protocols can be

  - hierarchical (e.g., Low Energy Adaptive Clustering Hierarchy (LEACH) [Heinzelman *et al.*, 2000], Threshold sensitive Energy Efficient sensor Network protocol (TEEN) [Manjeshwar and Agarwal, 2001], Adaptive Periodic Threshold sensitive Energy Efficient sensor Network protocol (APTEEN) [Manjeshwar and Agarwal, 2002], Zone Routing Protocol (ZRP) [Haas and Pearlman, 1998], Zone-based Hierarchical Link State routing (ZHLS) [Joa-Ng and Lu, 1999], Hybrid Ad hoc Routing Protocol (HARP) [Nikaein *et al.*, 2001]);
  - distance vector based (e.g., TinyOS beaconing [Hill *et al.*, 2000], TinyLUNAR [Osipov, 2007], Wireless Routing Protocol (WRP) [Murthy and Garcia-Luna-Aceves, 1996], Destination Sequence Distance Vector Routing protocol (DSDV) [Perkins and Bhagwat, 1994], DSR [Johnson and Maltz, 1996] and AODV [Perkins and Royer, 1999]);
  - link-state protocols (e.g., INSENS [Deng *et al.*, 2002], Optimized Link State Routing (OLSR) [Jacquet *et al.*, 2001]); or
  - data-centric (e.g., Directed Diffusion [Intanagonwiwata *et al.*, 2000]).

  In hierarchical protocols, the nodes form clusters, they elect a cluster leader, and forward data packets to the cluster leader, which then passes further the packets directly to other higher level cluster leaders, or to the destination.

  Distance vector protocols select the next hop towards the destination based on some distance-like routing metric. In TinyOS beaconing, for instance, a beacon message originating from the base station is flooded in the network, and each node chooses the node from which it first received the beacon as the next hop towards the base station. Thus, the time needed for the beacon to reach a node is used as the metric.

  Using link-state protocols, each node exchanges topology information with other nodes of the network, and thus, each individual node can reconstruct the topology and calculate routes in the network. In case of wireless sensor networks, link-state routing is often centralized, which means that sensor nodes send their link-state information to the base station, and based on these link-state information, the base station reconstructs the topology of the entire network and computes the routing tables for every node. The routing tables are then distributed to the nodes. The main drawback of this approach is that it does not scale well, and therefore, it cannot be applied in large networks.

  Finally, in the case of data-centric routing protocols, the next hop towards the destination is selected based on the content of the data packets. The advantage of these protocols is that the nodes do not need globally unique addresses, as routing decisions are not based on addressing information.

- *Location-based routing protocols*: These protocols (e.g., Greedy Perimeter Stateless Routing (GPSR) [Karp and Kung, 2000], Greedy Other Adaptive Face Routing (GOAFR) [Kuhn *et al.*, 2003], Distance Routing Effect Algorithm for Mobility (DREAM) [Basagnia *et al.*, 1998]) are also called position-based or geographic routing

protocols. Here each node forwards a packet based on the location of the destination, which is carried by the packet, and the locations of the forwarding node's neighbors. These protocols are often considered stateless, because the nodes do not need to store any additional routing information besides the locations of their neighbors. As a consequence, location-based routing protocols are mainly concerned with the message forwarding function of routing, and the discovery function is reduced to neighbor discovery instead of route discovery.

- *Hybrid protocols*: Hybrid protocols use both geographic and topological information to forward data packets (i.e., sensor nodes maintain some additional routing information besides the locations of their neighbors). These protocols are typically designed to incorporate energy-awareness in the simple forwarding process of geographic routing approaches (e.g., Geographic Energy Aware Routing (GEAR) [Yu *et al.*, 2001], Energy Aware Routing (EAR) [Shah and Rabaey, 2002]).

Another widespread method that is used to classify wireless network routing protocols is based on how routing information is retrieved during the route discovery and maintained by network nodes. Based on this, one can distinguish proactive, reactive, and hybrid protocols.

- *Proactive protocols:* Employing these protocols, all nodes continuously monitor links between nodes, and they attempt to maintain a consistent, up-to-date routing information. In particular, all nodes are required to maintain a consistent view of some part or all of the network topology, and when a change in this topology occurs, respective updates must be propagated to notify other nodes. In order to monitor topology changes, nodes proactively update network state and maintain a route regardless of whether data traffic exists or not. Thus, the overhead of maintaining up-to-date topology information is usually high. On the other hand, a source can calculate a path to a particular node faster than reactive protocols (see below) that is an advantage of these protocols. These protocols include WRP [Murthy and Garcia-Luna-Aceves, 1996], DSDV [Perkins and Bhagwat, 1994], DREAM [Basagnia *et al.*, 1998], or OLSR [Jacquet *et al.*, 2001].

- *Reactive protocols:* These protocols are also called on-demand protocols as a routing path is discovered only when it is needed. The route discovery procedure terminates either when a route has been found or when no route is available after the examination of all or some route permutations.

  As active routes may be disconnected due to node mobility, a route maintenance procedure is always provided to recover from route break-ups. Compared to proactive routing protocols, the control overhead is lower, and thus, reactive routing protocols have better scalability than proactive routing protocols in wireless networks. However, when using reactive routing protocols, source nodes may suffer from long delays for route discovery before they can forward data packets. DSR [Johnson and Maltz, 1996], AODV [Perkins and Royer, 1999], or TinyLUNAR [Osipov, 2007] are prominent examples for reactive routing protocols in wireless networks.

- *Hybrid protocols:* Hybrid routing protocols are proposed to combine the merits of both proactive and reactive routing protocols and overcome their shortcomings. In general, hybrid routing protocols for mobile ad hoc networks exploit hierarchical network architectures. Proper proactive and reactive routing approaches are used at different hierarchical levels, respectively. Hybrid routing protocols for mobile ad hoc networks include the ZRP [Haas and Pearlman, 1998], ZHLS [Joa-Ng and Lu, 1999] or HARP [Nikaein *et al.*, 2001] protocols.

In this dissertation, I consider reactive distance vector based protocols and a proactive link-state routing protocol.

2.1.2 Examples for routing protocols

This subsection briefly introduces the operation of some mainstream wireless routing protocols. Specifically, I present DSR, AODV, TinyOS beaconing, and TinyLUNAR. The security of these protocols is analysed later in this dissertation.

Dynamic Source Routing (DSR)

DSR [Johnson and Maltz, 1996] is a reactive routing protocol proposed for wireless ad hoc networks. The protocol has two mechanisms, route discovery and route maintenance, which work together to enable nodes to discover and maintain routes to arbitrary destinations in the network. We only consider route discovery as only its security is focused in this dissertation.

In the basic version of DSR, a source node initiates a route discovery towards a destination node by creating and broadcasting a Route Request (RREQ) message. This message contains the node identifiers of the source and the destination, a unique request identifier generated by the source, and a record listing the node identifiers of each intermediate node through which this particular copy of the RREQ message has been forwarded. The source initializes this record to an empty list.

A node receiving a RREQ message checks whether it has already received a request with the same request identifier or its own node identifier is listed in the route record of the RREQ message. If so, the receiver discards the message. Otherwise, the receiver checks whether it is the destination. In case this holds, the receiver replies with a Route Reply (RREP) message which contains the copy of the accumulated route record that is traversed by the received request. If the receiver is not the destination, it appends its node identifier to the accumulated route record in the request and rebroadcasts the request. In this way, the RREQ message floods the whole network.

When the destination receives an RREQ, it can check whether it already has a route in its route cache towards the source. If so, the destination can send the reply using that route. Otherwise, the destination can initiate a new route discovery towards the source by piggybacking the reply on a new route request message that is sent towards the source. Alternatively, the destination can simply reverse the route received in the request message and propagate the reply back on this reverse route, if the links are assumed to be bidirectional

in the network. The source receiving the reply can forward data messages towards the destination, where each data message contains the list of node identifiers. This list is received in the RREP message, and represents the route on which the message should be forwarded until it finally reaches its destination. If routes are ranked based on their length in terms of hop count, a reply is also sent by the destination when a request is received with the same request and source identifier, but it has the shortest route record among all received requests. Similarly, the source can later update its routing information if it learns of a better (shorter) route[1]. Note that DSR allows unidirectional links to be used when necessary, which makes the protocol usable in such networks where links can be asymmetric.

In an extended version of DSR, a node overhearing any packet may infer routing information from that packet and add this information to its own route cache. In particular, the route used in a data packet or the accumulated route record in a RREQ or in a RREP message can all be cached by any node. However, route caching must be used with care in case of unidirectional links and in order to avoid RREP storms. This is further discussed in [Johnson and Maltz, 1996].

Ad hoc On Demand Distance Vector (AODV) routing

AODV [Perkins and Royer, 1999] is another reactive routing protocol proposed for wireless ad hoc networks. Similarly to DSR, its route discovery part consists of two phases: the route request and route reply phase.

When the source wishes to send a data message towards the destination for which it has no routing information in its table, it forms a RREQ message and broadcasts that to its neighbors. This message contains the node identifiers of the source and destination, the broadcast identifier which uniquely identifies a request originated from the source, and a hop count value. This broadcast identifier is incremented when the source initiates a new request. If a node receiving a request has already received a request with the same source identifier and broadcast identifier, then the request is discarded. Otherwise, the node checks whether it is the destination. If not, the node stores the source and destination identifiers and the broadcast identifier along with the next-hop id from which the RREQ is received in its routing table, increments the hop count value in the request, and rebroadcasts the request. In this way, all nodes who receive the RREQ can set up a reverse path towards the source. These reverse path entries should be maintained until the reception of the corresponding reply message coming from the destination.

When the destination receives an RREQ message, it checks whether this RREQ message contains smaller hop count value than the requests received so far from the source with the same broadcast identifier. If so, or if it is the first RREQ that is received with that broadcast id, the destination sends an RREP message back to the source, which contains the source and destination identifiers. Otherwise, the destination discards the message. This reply message is directly sent to the neighbor from which the corresponding request message is received. Before

[1]In that case, similarly to AODV, the request identifier should be incremented rather than generated randomly when the source initiates a new request.

forwarding the reply back towards the source node, all intermediate nodes set a routing entry towards the destination, where the next-hop towards the destination is the neighbor from which the reply is received. A node who receives an RREQ message but does not receive any RREP messages purges the routing entry set towards the source after a specified time.

The source node can begin data transmission as soon as the first RREP is received and can later update its routing information if it learns of a better route (i.e., it has a smaller hop count value).

AODV also uses source and destination sequence numbers in the request and reply messages in order to implement caching mechanisms, to provide loop-free property, and to handle link breakage (e.g., due to node mobility). The caching mechanism enables each intermediate node to send a reply to a particular request immediately, if it knows a fresher route towards the destination than the source of that request does. This caching mechanism is further detailed in [Perkins and Royer, 1999].

TinyOS beaconing

Originally, the authors of TinyOS [Hill *et al.*, 2000] proposed a very simple routing protocol, called TinyOS beaconing. In this protocol, each node is addressed by a globally unique identifier, and the base station periodically initiates a route discovery by flooding the network with a beacon message. Upon the reception of the first beacon within a single beaconing interval, each sensor node stores the identifier of the immediate sender of the beacon as its parent (a.k.a., next-hop towards the base station), and then re-broadcasts the beacon after replacing the sender identifier with its own identifier. As for each node only one parent is stored, the resulted routing topology is a tree. In the data forwarding process, every sensor node receiving a data packet forwards that towards the base station by sending the packet to its parent. This beaconing mechanism is a straightforward method to build a simple routing topology, where each node sets a neighbor as its parent if this neighbor lies on the fastest path to the base station. The protocol assumes symmetric links in the network and does not consider any energy metric to optimize network lifetime.

Tiny Lightweight Underlay Adhoc Routing for Wireless Sensor Networks (TinyLUNAR)

TinyLUNAR [Osipov, 2007] is a topology-based routing protocol proposed for wireless sensor networks that also supports point-to-point communication between individual sensor nodes. Using the label-switching routing paradigm, TinyLUNAR has only one byte addressing overhead per packet in the data forwarding phase, which, considering the high communication costs in wireless environment, makes it an efficient routing scheme in relatively static networks. Similar to the previously described AODV and DSR, TinyLUNAR also floods the network with an RREQ message to discover routes towards the destination. The destination can initiate the built of a reverse route towards the destination by replying to this request. TinyLUNAR assumes bidirectional links between sensor nodes.

Route Request: A source node S initiates the route discovery to destination D by flooding the network with a route request message:

$$S \rightarrow * : (\mathsf{RREQ}, \mathsf{rnd}, S, D, addr_S, label^{\mathsf{In}}_{S \rightarrow S})$$

where rnd is a randomly generated request id, $label^{\mathsf{In}}_{S \rightarrow S}$ is the incoming label of S towards S, and $addr_S$ is the locally unique network address (e.g., MAC address) of S. In fact, $label^{\mathsf{In}}_{S \rightarrow S}$ is a memory address inside the routing table of S and contains an application identifier which originally initiated the route discovery process.

A node J receiving this broadcast message checks whether it has received the request earlier based on rnd, S, and D. If so, J silently drops the request. Otherwise, J stores the quadruple $(addr_S, label^{\mathsf{In}}_{S \rightarrow S}, \mathsf{rnd}, \mathsf{lifetime})$ in its routing table, where $\mathsf{lifetime}$ is set to a predefined value $\mathsf{MaxLifetime}$ and $addr_S$ is the local network address of the neighboring node from which the request is received. The value of $\mathsf{lifetime}$ is periodically decremented when the routing table entry is not used. If it reaches the value of zero, then the entry is purged from the routing table. At the same time, each time the entry is used, the value of lifetime is reset to $\mathsf{MaxLifetime}$. Using this entry, J can forward messages to S. Afterwards, J broadcasts the message as follows:

$$J \rightarrow * : (\mathsf{RREQ}, \mathsf{rnd}, S, D, addr_J, label^{\mathsf{In}}_{J \rightarrow S})$$

where $addr_J$ is the locally unique network address of J, and $label^{\mathsf{In}}_{J \rightarrow S}$ is the incoming label of J towards S. Essentially, $label^{\mathsf{In}}_{J \rightarrow S}$ is the local memory address of the routing entry where J stores the corresponding entry pointing to S (i.e., this entry contains the quintuple S, $addr_S$, rnd, $label^{\mathsf{In}}_{S \rightarrow S}$, and $\mathsf{lifetime}$). A node receiving this request performs the same operations that J did, and thus, it can forward messages to S through J afterwards. Note that nodes do not store the globally unique network id of the next-hop towards S, as these next hops are addressed by the locally unique network addresses which are included in the header of each sent message by default.

After the network is flooded, each node that received the request has an entry set towards S. In this way, the *backward traffic flow* is constructed which is defined by the set of all routing entries created at intermediate nodes. This traffic flow is associated with S at the endpoint D.

Route Reply: When destination D receives the first request message, for instance from node Z, it creates a routing entry similar to all nodes who receive the request. After that, D sends a reply to S:

$$D \rightarrow Z : (\mathsf{RREP}, \mathsf{rnd}, addr_D, label^{\mathsf{Out}}_{Z \rightarrow S}, label^{\mathsf{In}}_{D \rightarrow D})$$

where rnd is the random identifier of the corresponding request originated from S, $label^{\mathsf{Out}}_{Z \rightarrow S}$ is the incoming label of Z towards S (i.e., the outgoing label of D towards S) received in the request, and $label^{\mathsf{In}}_{D \rightarrow D}$ is the incoming label of D. Here, $label^{\mathsf{In}}_{D \rightarrow D}$ is a memory address inside the routing table of D and, similarly to S, contains an application identifier which originally initiated the route discovery process. Note that Z is addressed by its incoming label and its local network address, which is included in the message header and not listed in the message

content. When Z receives the reply, it first creates a routing entry set towards D. This entry contains $addr_D$, rnd, and $label^{\mathsf{In}}_{D\to D}$, where $addr_D$ is the local network address of the neighboring node from which the reply is received. From now on, Z can forward messages to D. Then, Z looks up the entry addressed by $label^{\mathsf{Out}}_{Z\to S}$ in its memory (routing table), and forwards the message to the node contained by this entry. Let us assume that Z received the corresponding request from node K first. Then, Z sends the following message to K:

$$Z \to K : (\mathsf{RREP}, \mathsf{rnd}, addr_Z, label^{\mathsf{Out}}_{K\to S}, label^{\mathsf{In}}_{Z\to D})$$

K performs the same steps that Z did, and forwards the reply to the next node whose address is retrieved from the entry at memory address $label^{\mathsf{Out}}_{K\to S}$.

All subsequent nodes receiving the reply do the same operations that Z did. In this way, the *forward traffic flow* is constructed, which is defined by the set of all routing entries created at intermediate nodes. This traffic flow is associated with S at the endpoint D. Finally, after S receives the reply, it can send data messages to D.

Route Request optimization: Intermediate nodes receiving a control message can forward messages between the source/destination nodes, but they cannot send messages to them or any other nodes using the same traffic flow. In order to create a separate traffic flow between an intermediate node and an endpoint, the intermediate node must initiate a new route discovery by sending a request message towards the endpoint. Note that this request does not need to be broadcast, as the existing traffic flow between the source/destination pair can be used to forward the new request towards the intended endpoint. In order to indicate the proper actions to be taken to the intermediate nodes, this type of request is distinguished from the ordinary request message by its message type identifier in the packet header.

Data forwarding: Each node receiving a data packet can determine the next hop by looking up the routing entry addressed by the incoming label retrieved from the packet. Then, the node can update the incoming label in the packet with the outgoing label found in the routing entry. Note that intermediate nodes between endpoints S and D do not need to be aware of identities S and D. All data packets sent between S and D contain the incoming label of the next node on the route, and do not need to include further network addresses besides the address of the next node. As labels have size of 1 byte, TinyLUNAR has only 1 byte addressing overhead per data message which makes it an effective routing mechanism in wireless sensor networks where nodes are stationary or show moderate mobility during their operation.

2.1.3 Objectives of attacks

Generally speaking, the adversary primarily intends to thwart the objectives of routing protocols. More specifically, it wants to degrade the performance of routing, or ultimately, it may attempt to completely disrupt the routing service and cause network malfunction. Degrading the performance of routing can mean degrading the packet delivery ratio, shortening the network lifetime, and/or increasing the network delay. In addition to these, the objective of attacking the routing protocol can be to increase the hostile control over the traffic. Note that

some of these adversarial goals are highly correlated (e.g., if the adversary can successfully divert the traffic through adversarial nodes, then it can easily degrade the packet delivery ratio or increase the network delay by dropping packets and delaying packet forwarding). Also note that shortening the network lifetime is a more typical objective of attacks in sensor than in other wireless networks due to the more limited power supply of sensor nodes.

Informally, the task of secure routing is to fulfil the objectives of routing protocols even in the presence of an adversary, who primarily intends to thwart the objectives of routing protocols. This can be achieved by preventing attacks, or detecting and then eliminating them. If elimination is not viable (i.e., that would be too costly), a secure routing protocol should mitigate the effects of attacks and attempt to recover the routing service. All techniques that prevent or detect and eliminate attacks are called countermeasures. In order to develop appropriate countermeasures, one has to first map the capabilities of the adversary, which is shortly called the adversary model. The adversary model also includes the basic attack methods that it employs to subvert the routing service.

2.1.4 Capabilities of the adversary

The adversary has control over some nodes in the network that are further called *adversarial nodes*. This control is gained by either deploying new corrupted nodes or by taking the control over honest nodes using some malicious software (e.g., worms or viruses). In case of sensor networks, adversarial nodes can be sensor-class devices and more powerful laptop-class devices. It is quite reasonable to assume that both sensor-class and laptop-class devices can be easily acquired by the adversary, or alternatively, it can capture honest sensor-class devices directly in the network field. Of course, capturing is viable only if sensor nodes are not tamper resistant devices and the adversary can gain unsupervised access to them. Sensor-class devices have identical capabilities to an ordinary sensor node (i.e., their energy supply as well as their computational power is typically heavily constrained). On the contrary, laptop-class devices are more powerful; besides having unconstrained energy supply and computational capability, they also have powerful transmitters with much greater power range than sensor nodes have. Additionally, laptop-class devices may also have more sensitive receivers, though such equipment bears much higher costs than transmitters.

In the models proposed so far, the adversary has full control over the communications of the honest protocol participants. This means that it can read, modify, or delete any of the messages sent between any protocol participants, and it can also inject forged messages to any protocol participant. This may be an appropriate model in Internet-like networks, where having access to some special network elements, such as routers, allows the adversary to have this level of control. This is due to the typical hierarchical structure of these networks. On the other hand, in most wireless networks, an adversary can have a similar level of control over the communications only if it is physically present everywhere, which is due to the more typical flat structure of wireless ad hoc and sensor networks. Although this continuous physical presence of the adversary can hold for wireless ad hoc and even more for wireless sensor networks, it is considered to be very costly, and hence, unrealistic in many applications. Note that wireless

networks can also exhibit hierarchical structures like a star-like traditional wired network, but this is not general due to its lower fault tolerance[2] or the random deployment of wireless nodes. Therefore, I assume that the adversary has communication capabilities comparable to those of an average node in the ad hoc network. This means that an adversary can hear only those messages that were transmitted by neighboring nodes, and similarly, the transmissions of the adversary are heard only by its neighbors. This does not necessarily mean that wireless networks are easier to attack than wired networks. I just want to point out that the adversary models in the two cases are different. I assume that the number of the overheard honest nodes is typically less than the number of all honest nodes in the network.

The adversary is active in the sense that besides eavesdropping messages it can fabricate and insert new messages in transit, and in addition, it can modify, delete, re-order and delay existing messages that traverse her without following the routing protocol rules faithfully. Before investigating more sophisticated attacks that employ the previously listed message manipulations, I describe how the adversary can perform such message manipulations.

Injection of messages in a radio channel is trivial. Message deletion can also be easily done by simply not forwarding a message according to the protocol rules, or by performing jamming [Xu *et al.*, 2005]. Message modifications and re-ordering can be performed in a straightforward way if the adversary acts as a relay node between the sender and the receiver (i.e., the sender and the receiver cannot reach each other directly). However, if the receiver and the sender can communicate directly, then the adversary must use sophisticated jamming techniques that prevent the receiver from receiving messages, while at the same time allow the adversary to receive those messages. Once a message is deleted in this way, the adversary can modify it and send the modified message to the receiver. In particular, message modification is only feasible, if both the sender and the receiver nodes are within the communication range of an adversarial node.

Here, I sketch two scenarios for message modification, which are illustrated in Figure 2.1. By these simple examples, I intend to point out the feasibility of message modification assuming even direct communication between the sender and the receiver node. I further assume that communication range implies interference range, and vice-versa.

Scenario 1: There are two honest nodes X and Y, and node X intends to send a message m to node Y. A_1 and A_2 are adversarial nodes, where A_2 is able to interfere with Y's communication, but not with X's and A_1's communication. Let A_1 be in the communication range of X and Y, whereas A_2 can only communicate with Y. When X transmits m to Y, node A_1 overhears m, meanwhile A_2 performs jamming to cause Y not to be able to receive m. In order to take this action, A_1 and A_2 are connected by an out-of-band channel; thus, A_1 can send a signal to A_2 when A_2 should start jamming Y's communication. It is also feasible that A_2 performs constant jamming for a certain amount of time; afterwards, A_1 can send the modified message m' to Y.

Scenario 2: In this scenario, there is only one adversarial node denoted by A. We

[2]Note that most wireless ad hoc and even sensor networks are deployed in harsh environments, and thus, wireless nodes can become out-of-order easier than physically protected wired routers.

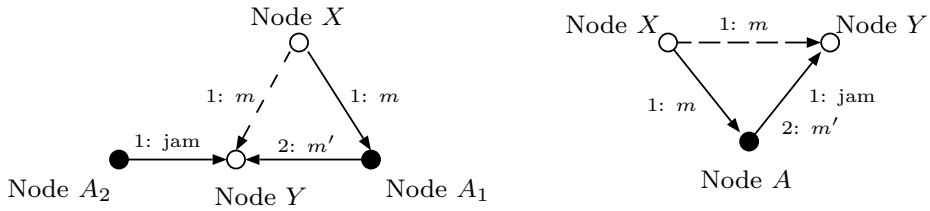


Figure 2.1: Message modification performed by the cooperation of two adversarial nodes A_1 and A_2 (on the right-hand side) in Scenario 1, and employing overhearing, jamming, and relaying with a single adversarial node A (on the left-hand side) in Scenario 2. Honest nodes are labelled by X and Y. Arrows between nodes illustrate the direction of communication, the sequence of message exchanges are also depicted on these arrows. Dashed arrows illustrate failed message delivery caused by jamming.

assume that transmitting a message from the routing sublayer consists of passing the message to the data-link layer, which, after processing the message, also passes it further to the physical layer. The data-link layer uses CRC in order to provide some protection against faults in noisy channels; a sender generally appends a frame check sequence to each frame (e.g., see [(IEEE), 2003]). The adversary can exploit this CRC mechanism to modify a message in the following way (illustrated on Figure 2.1). When X transmits message m to Y, node A also overhears m, in particular, he can see the frame(s) belonging to m. A intends to modify message m. Here, we must note that most messages originated from the routing sublayer are composed of only one frame per message in the data-link layer due to performance reasons, especially when they are used to discover routing topology. Upon reception of the frame corresponding to the message, the adversary can corrupt the frame check sequence by jamming once the data field of the frame has been received. This causes node Y to drop the frame (and the message), since Y detects that the last frame is incorrect, and waits for retransmission. At this point, if some acknowledgement mechanism is in use, A should send an acknowledgement to X so that it does not re-send the original frame. In addition, A retransmits message m' in the name of X, where m' is the modified message.

Note that even if the receiver and the sender can communicate directly without adversarial relaying, these modification attacks can be implemented, which is a clear distinction compared to traditional wired networks.

The feasibility of jamming attacks is studied and demonstrated in [Xu *et al.*, 2005]. Although, the authors conclude in that paper that the success of jamming attacks mainly depends on the distance of the honest nodes and the jammer node, various jamming techniques have been presented there that can severely interfere with the normal operation of the network.

Finally, I distinguish two types of adversaries. An *outsider adversary* is assumed to be able to manipulate messages sent by honest nodes, however, it cannot control legitimate nodes. On the contrary, an *insider adversary* has all the power as the outsider adversary, and additionally, it is able to control some legitimate nodes in the routing process (this may also mean the compromise of the cryptographic keys of those nodes).

2.1.5 Attack methods

Although the focus of this dissertation is the security of the route discovery part of wireless routing protocols, in this subsection, I discuss the main attack methods against both route discovery and data forwarding.

The simplest attack methods are composed of the basic message manipulation techniques. These include *dropping*, *modification*, *delaying*, *injection* and *re-ordering* of routing control messages. In order to further refine these methods, we separate the route discovery and the data forwarding phase of routing protocols. In both the route discovery and data forwarding processes, the adversary can inject extra packets in order to decrease the throughput of the network and to consume valuable network resources in wireless sensor networks (leading to denial-of-service). In the route discovery phase, injecting a forged control packet can result in corrupt routing states at honest nodes that may ultimately yield increased traffic control as well as shortened network lifetime and increased network delay. Dropping control packets has trivial effects: in this way, the adversary can separate some of the nodes from the destination (or the base station in sensor networks) that can only reach the destination through an adversarial node. The adversary can also degrade the packet delivery ratio and the network delay by dropping packets in the data forwarding process. By modifying control packets, the adversary can cause honest nodes to store corrupt routing states, which may have similar effects to injecting forged control packets. Furthermore, the adversary can also modify data packets, which may lead to re-transmissions, and hence increased energy consumption. Re-ordering and delaying of control packets can influence the next-hop selection mechanism.

Besides these basic packet manipulation attacks, the adversary may also be capable to mount attacks at higher level. These are *tunnelling*, *rushing*, *selective forwarding*, and *replay* attacks.

In the tunnelling attack, the adversary controls some corrupted nodes in the network, and tunnels routing control messages between these controlled nodes in the payload part of normal data packets using the multi-hop forwarding mechanism of the network. In this way, the adversary can make some routes appear shorter than they really are, and thus, these routes may be preferred by the other nodes.

Rushing [Hu, 2003] is a protocol-dependant attack that is also a threat against some sensor network routing protocols. In particular, the adversary can launch this attack only against routing protocols that employ a duplicate suppression technique to control flooding. When using duplicate suppression, a node only considers the first copy of a given control packet and drops any further copies. For instance, a node A running TinyOS beaconing (as it is described in Subsection 2.1.2) sets the neighboring node from which it received the first copy of the beacon as the next hop towards the base station. Any further beacons of the same beaconing period are simply discarded by A. The adversary employing rushing can exploit this duplicate suppression technique to divert the traffic: The adversary forges a beacon and broadcasts that to node A. As a result, A will set the identifier found in this forged beacon as the next-hop towards the base station. Later, when A receives the real beacon, it discards it due to duplicate suppression. In this way, the adversary can divert traffic to itself and

increase hostile traffic control. This can be the first step of further severe attacks.

Selective forwarding refers to the capability of dropping data packets in the data forwarding process in a selective manner. Generally, this attack can be a second step after a successful tunnelling or rushing attack. If the adversary jointly uses selective forwarding with any route diversion techniques, then it can easily setup a *blackhole* (where all packets are dropped) or a *grayhole* (where only specific packets are dropped).

Replay attacks can occur during topology construction as well as during the data forwarding process. The adversary can replay obsolete routing control packets that no longer reflect the current network topology, which may yield inefficient routing paths. In the data forwarding process, replaying obsolete data packets causes incorrect reports about the monitored environment in sensor networks.

I additionally note that while topology-based and link-state routing protocols are usually vulnerable to control packet manipulation, geographic and hybrid routing protocols seem to be more resistant against these attacks.

In the sequel of this subsection, I review additional attacks that are mainly related to neighbor discovery, such as the *wormhole attack*, the *Sybil attack*, and the *node replication attack*. I also consider these attacks against routing, since many wireless routing protocols integrate neighbor discovery as part of a cross-layer design.

A wormhole is an out-of-band connection, controlled by the adversary, between two physical locations in the network. The adversary installs radio transceivers at both ends of the wormhole, and it transfers packets (possibly selectively) received from the network at one end of the wormhole to the other end via the out-of-band connection, and re-injects the packets there into the network.

The effect of a wormhole on neighbor discovery is that some nodes that would not be neighbors otherwise may establish a neighbor relationship. This has a direct effect on route discovery mechanisms that operate on the connectivity graph, since they may identify routes that use virtual links created by the adversary. Thus, a well placed wormhole gives considerable power to the adversary, who can monitor the network traffic flowing through the wormhole, or mount a black hole attack by permanently or selectively dropping data packets sent via the wormhole.

Some routing protocols do not rely on explicit neighbor discovery mechanisms, but the nodes discover their neighbors implicitly via processing the overheard routing control messages. Many of these protocols are equally vulnerable to the wormhole attack. For instance, an adversary can use a wormhole to mount a rushing attack against routing protocols based on flooding a route request and controlling the flood with duplicate suppression.

The wormhole attack has similar effects on routing protocols than the tunnelling attack, but it is based on slightly different assumptions about the adversary. In particular, in the tunnelling attack, the adversary controls some corrupted nodes in the network, and tunnels routing messages between these controlled nodes in the payload part of normal data packets using the multi-hop forwarding mechanism of the network. Therefore, by definition, in order to mount a tunnelling attack, the adversary needs to have corrupted nodes in the network, which use (possibly compromised) identifiers. In contrast to this, the adversary can mount a

wormhole attack without corrupting any nodes or compromising any node identifiers, because the wormhole uses only low level repeaters transparent to higher layer protocols.

In a Sybil attack [Douceur, 2002], a single adversarial node illegitimately uses *multiple* identities during the routing process. This can have devastating effects on multipath routing protocols [Karlof and Wagner, 2003], because a node may believe that it routes packets via node disjoint paths, while in reality these paths may all go through the adversarial node implementing the Sybil attack. Sybil attacks employed together with tunnelling or wormhole attacks can be even more powerful, as the tunnels and the wormholes can be used by the adversarial nodes to share their invented identities.

The node replication attack [Bryan *et al.*, 2005] is the dual of the Sybil attack, where the adversary uses the *same* identity for multiple devices, and thus a single adversarial node may be virtually represented in multiple locations in the network. Replication attacks can also severely influence the operation of most routing protocols; in the worst case, the adversary can copy the identity of the base station and use it in different locations of the network. If the adversary manages to impersonate the base station, then it may be able to attract all traffic to it; this is often referred to as the *sinkhole* attack [Karlof and Wagner, 2003].

All these basic attack methods listed so far can serve as building blocks for further more complex attacks such as the HELLO flood attack against TinyOS beaconing, the creation of routing loops, black- and grayhole attacks, or route diversion attacks [Karlof and Wagner, 2003].

2.1.6 Examples for attacks

In this subsection, I illustrate some of the previously described attack methods on DSR, AODV, Directed Diffusion, GPSR, TinyOS beaconing, and TinyLUNAR, where DSR, AODV, and GPSR are primarily proposed for ad hoc networks, while the others are typical sensor network routing protocols.

DSR

In the lack of any security mechanisms, DSR is subject to several simple attacks described in the previous subsection. For instance, the adversary can cause honest nodes to store non-existing routes in their route cache by inserting new node identifiers in the route record of a RREQ message. Even more this attack may not be detected by neighboring nodes, if they are not aware of every two-hop neighbors. In Figure 2.2, such an attack is illustrated. Source S initiates a route discovery towards D. Adversarial node A forwards the request faithfully. However, when the reply comes back on the route U, T, A from destination D, A will replace identifiers T, U with B in the route record of the reply message. Hence, S will believe that there is a path S, B, D, however, there is no such route in the network. Note that S cannot detect the misdeed as B is a neighbor S. Moreover, there is another route S, W, X, D that is discarded at S due to its larger size.

As it can be seen, the adversary can also remove node identifiers from the route record without being detected. Even more, shortening the route record increases the probability

of accepting this non-existing route at the destination for data forwarding in more complex networks where there are several routes between the source and the destination without any adversarial nodes lying on them. More generally, a "cut-and-paste" attack also works here: the adversary can always replace any parts of the accumulated route record with any (but preferably shorter) sequence of node identifiers.

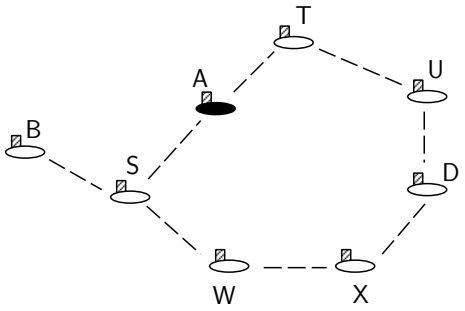


Figure 2.2: Message manipulation attacks against DSR and AODV. Honest nodes are denoted by S, B, X, Y, W, T, U and D. The adversarial node is A. Dashed arrows denote bidirectional links between nodes.

AODV

Similar to DSR, AODV does not use any security protection either; thus, it is also vulnerable to several simple attacks. Although AODV does not accumulate node identifiers in the control messages, the adversary can still initiate fake route discoveries in the name of honest nodes and can reply with forged RREP messages. For instance, in Figure 2.2, source S initiates a route discovery towards destination D. The adversary can achieve that honest node S sets an entry towards destination D along with neighbor B from which, in turn, there is no route to D. In order to implement this attack, adversarial node A forwards the request towards D according to the AODV rules. However, when the reply comes back from D, A sends that (or a request that is sent by D to discover S) in the name of B to S. As a result, S believes that there is a route between B and D, and forwards all data packets through B to D. Note that the reply received from W at S will be dropped, if the adversary additionally decreases the hop count in the reply.

In general, the adversary can always decrease the hop count value in the control message, which causes the source or the destination to accept those routes on which the adversarial node lies. The malicious manipulation of the destination and source sequence numbers can cause honest nodes to accept obsolete control messages that no longer reflect the current network topology.

Directed Diffusion

Directed Diffusion [Intanagonwiwata *et al.*, 2000] is another mainstream topology-based routing protocol for wireless sensor networks. The base station initially floods the network with

an *interest*, which contains attribute-value pairs describing the requested data. Upon the reception of an interest, each sensor node sets a gradient pointing to the immediate sender node. A gradient defines the requested data at each sensor node in conjunction with the next-hop towards the base station through which a message containing the requested data should be forwarded. Moreover, each gradient is weighted proportionally to the amount of data that is allowed to traverse the gradient. If a node receives the same interest from different neighbors, then the node can set multiple gradients, which correspond to the same interest, pointing to different neighbors. The neighbors are differentiated by locally unique identifiers.

The data is forwarded to the base station by the intermediate nodes along their gradients. If there are more gradients at a node for the same interest, then the node forwards one copy of the message along each gradient. After a while, the base station selects the route with the best quality and increases the weight of the gradients along the route (positive reinforcement), whereas it decreases the weights on the others (negative reinforcement).

Intermediate nodes may aggregate the received data, and forward this aggregated data along the corresponding gradients at a rate that is proportional to the weight of the gradient. The base station periodically re-sends the interests along the used routes in order to keep the gradients of intermediate nodes alive. In this way, the base station keeps the empirically best routes and eliminates the routes that have worse quality. Optionally, all nodes can cache data in order to achieve shorter response time and increase robustness. A more comprehensive description can be found in [Intanagonwiwata *et al.*, 2000].

The adversary can easily mount black- and grayhole attacks against Directed Diffusion. A blackhole attack means that the adversary allures all traffic from a particular area along an adversarial node, and then drops all received packets. A grayhole attack is a more sophisticated selective forwarding; the adversary first allures the traffic and then selectively drops some data packets. Let us consider the network topology depicted in Figure 2.3. The adversarial node can simply allure the traffic by broadcasting a forged interest in the name of B. Thus, all nodes receiving this forged interest will send data packets to node A. A more clever adversary can exploit the reinforcement strategy of Directed Diffusion; the adversarial node A reinforces some paths without forging any interests (i.e., receiving the original interest from node B, A rebroadcasts that containing increased data-rate values). Consequently, all nodes receiving this modified interest will forward data packets towards the base station along A at higher data rates, which then can drop, modify, or forward packets at her own wish. Moreover, this false reinforcement also causes honest nodes' batteries to deplete faster.

GPSR

Greedy Perimeter Stateless Routing (GPSR) [Karp and Kung, 2000] is a geographic routing protocol proposed for wireless ad hoc and sensor networks that can be used to route data packets between any pair of nodes (i.e., it supports node-to-node communication). GPSR assumes that every node is aware of its own location and the locations of its neighbors.

Initially, nodes construct a planar subgraph based on the network topology in a distributive manner. This distributive planarization algorithm can be Gabriel Graph (GG) [Gabriel and

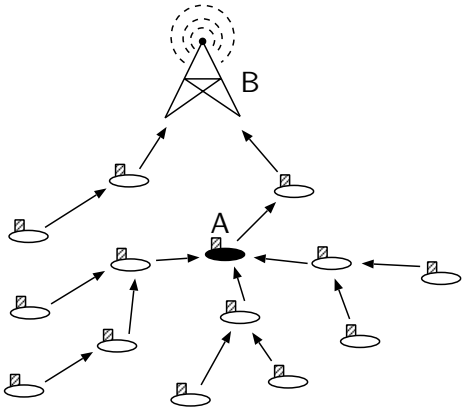


Figure 2.3: Black- and grayhole attack against Directed Diffusion. The only adversarial node is denoted by A. For each node, the solid lines denote the gradients set towards the base station.

Sokal, 1969], Relative Neighborhood Graph (RNG) [Toussaint, 1980], Crossing Link Detection Protocol (CLDP) [Kim *et al.*, 2005] or Lazy Cross Link Removal (LCLR) [Kim *et al.*, 2006]. We do not detail the planarization process further, more interested readers are referred to the corresponding literature. This planar graph will be used to circumvent voids in data forwarding.

Upon the reception of a data packet that carries the location of the destination node D, each node I checks whether it has a neighbor that is closer to node D than itself. If it has, the message is forwarded to that node. Otherwise, I switches to face routing mode, which means that it determines the neighboring face in the planar subgraph that is intersected by the imaginary line connecting I and the destination, denoted by d. After putting its own location into the packet, I uses the right-hand rule to select the next-hop on the perimeter of that face. Each node on the perimeter of the face uses the same rule to select the next-hop for the packet until one of the following events occurs:

- A node is reached that is either D or it is closer to D than I. In the latter case, the node that is closer to D than I performs the same steps that I did and the process repeats.
- An edge is reached, denoted by e, which is intersected by line d. In that case, GPSR switches to the neighboring face that is intersected by d (i.e., the neighboring face contains e).
- If the packet completely traverses the perimeter of the face (i.e., e or I is traversed again) without reaching a node being closer to D, then the packet is marked as undeliverable.

I show how the adversary can divert the traffic and create detours between a source and a destination causing increased energy consumption, and thus decreased network lifetime. In Figure 2.4, a source node S is assumed to send a packet to the base station B. As node E is closer to B than any other neighbors of S, S forwards the packet to E. Similarly, E forwards the packet to the first adversarial node A. In order to divert the traffic, A alters the destination

location in the packet to the location of the second adversarial node A'. Hence, following the GPSR rules the packet will be forwarded to A' along nodes H, K. Afterwards, A' recovers the original destination location to B's location, and the packet will be delivered along node Q. Therefore, the packet reaches node B along nodes $\mathsf{E}, \mathsf{A}, \mathsf{H}, \mathsf{K}, \mathsf{A}', \mathsf{Q}$ instead of nodes $\mathsf{E}, \mathsf{A}, \mathsf{C}$ that would be a much shorter and less energy consuming route.

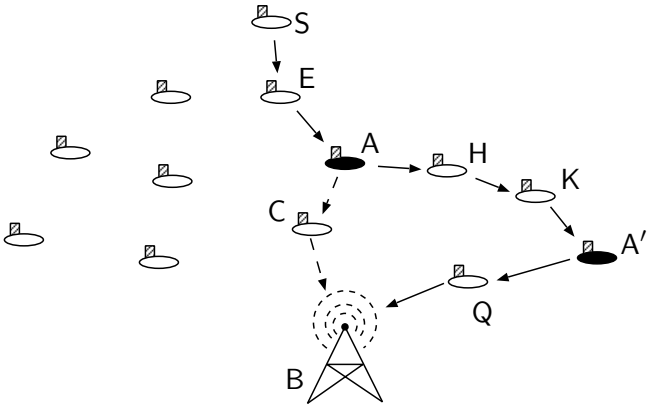


Figure 2.4: Route diversion attack against GPSR. The adversarial nodes are denoted by A and A'. The packet reaches B along nodes $\mathsf{E}, \mathsf{A}, \mathsf{H}, \mathsf{K}, \mathsf{A}', \mathsf{Q}$ that is denoted by solid arrows. In contrast to this, if the adversary does not divert the packet, it will traverse nodes $\mathsf{E}, \mathsf{A}, \mathsf{C}$, denoted by dashed arrows, which is a much shorter route.

TinyOS beaconing

In the following, I show how the adversary described in Subsection 2.1.4 can create a routing loop in a sensor network using TinyOS beaconing. Let us consider Figure 2.5. First, the base station B floods the network with a beacon containing its identifier. Before re-broadcasting the beacon, node E replaces the sender identifier with its own identifier to indicate to its neighbors that they can reach the base station through E. Receiving this beacon, the adversarial node A does not replace the sender identifier with A according to the protocol rules, but it replaces it with D. Hence, C sets D as its parent node, and rebroadcasts the packet with its own identifier causing node D to set C as its parent node. As a result, node C will forward all data packets to D and D will forward all data packets to C without ever reaching the base station and consuming valuable resources.

TinyLUNAR

In the following, I argue that impersonation attacks cause incorrect routing entries in TinyLUNAR.

Source impersonation: The adversary can use any honest node identifier as the source identifier of any request messages. For instance, in Figure 2.6a, if adversarial node A sends a forged request to node D, where the request contains S as the origin of the message, then D sets an entry towards S with next-hop identifier T. However, a packet sent to T cannot be

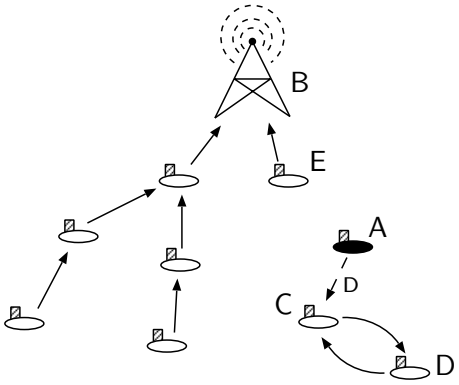


Figure 2.5: Creating a routing loop in TinyOS beaconing. The only adversarial node is denoted by A. For each node, the solid line points to the parent node. Adversarial node A rebroadcasts the beacon in the name of node D that is denoted by a dashed arrow. As a result, node C will believe that the sender of this beacon is node D. Thus, C sets D as its parent node.

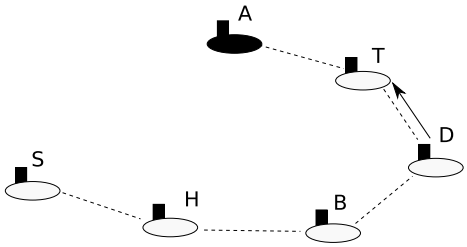
delivered to S.

Destination impersonation: The adversary can generate reply messages in the name of any honest nodes. For instance, let us assume in Figure 2.6b that S floods the network with a request in order to discover a route towards D. This request is also received by adversarial node A. Thus, A can generate a reply message in the name of D, which causes incorrect entry at node S, as this forged reply is likely to be received by S sooner than the untampered reply coming from S.

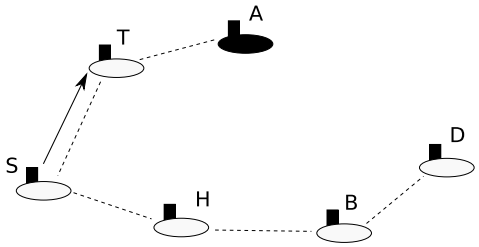
Neighbor impersonation: In Figure 2.6c, we illustrate neighbor impersonation attack. The adversarial nodes are A and A'. Assume that H can only be reached by S and A', and the adversary is aware of all nodes' identities and the local addresses of the nodes that it can reach (i.e., local addresses of H, S, B). Furthermore, S wishes to discover a path to D. First, S floods the network with a request which is received by adversarial node A. A rebroadcasts the request faithfully. However, when the corresponding reply comes back from D, A rebroadcasts that in the name of H (i.e., A uses H's identity and local network address, which is catched by A'). Finally, receiving this forged reply, S believes that D can be reached through H. However, as H does not receive any replies, it will not forward any messages towards D.

2.1.7 Considered attacks

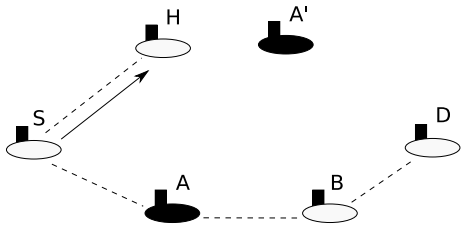
In this dissertation, I do not deal with all attacks described in Subsection 2.1.5 further. Particularly, *I focus on the basic message manipulation attacks that aim to corrupt the routing entries of honest nodes* where a routing entry is a representation of a route towards a particular destination node. This representation can be the list of identifiers of nodes constituting the route like in DSR, or the identifier of the next-hop along which there should be a route to the



(a) Source impersonation attack. D believes that S can be reached through T, however, there is no route between T and S.



(b) Destination impersonation attack. Here, S falsely sets an entry towards D through T, but there is no route between T and D.



(c) Neighbor impersonation attack. If A' does not forward the reply coming from D to H, but it does towards S, then S will believe that D is reachable through H. However, H does not have any entry towards D.

Figure 2.6: Impersonation attacks against TinyLUNAR. Dashed lines denote the neighborhood relations, whereas arrows denote the routing entries.

destination with a certain cost. The goal of the adversary is to cause honest nodes to store such routing entries that are not consistent with the underlying network topology, where the definition of consistency is protocol dependant; such routing entries are called as *incorrect routing entries*. For instance, in case of source routing, an incorrect entry can be a sequence of node identifiers, where some pairs of consecutive nodes in the sequence are not connected (i.e., the sequence cannot be represented by a physical route in the network topology). In the case of distance vector routing protocols, an incorrect entry contains a next-hop that is not a neighbor, or it is a neighbor but there is no route with the stored cost between that neighbor and the destination node in the network.

As the adversary pollutes the set of routing entries in honest nodes with incorrect entries, I further refer to these attacks as *routing state pollution attacks*. A routing state pollution attack has already been informally exemplified in Subsection 2.1.6. In this dissertation, *I focus on the prevention of routing state pollution attacks.*

As a direct consequence, I only investigate those routing protocols in the sequel that can be vulnerable to routing state pollution attacks (i.e., nodes locally store the representation of routes). The security of the rest of the routing protocols (like location based routing protocols) is beyond the scope of this dissertation.

Although the set of considered attacks is only a small subset of all attack methods, as it can be seen in the next chapters, even some of the existing "secure" routing protocols are vulnerable to routing state pollution attacks where the adversary uses only simple message manipulation techniques. I will demonstrate that routing state pollution attacks against wireless routing

protocols can be very subtle, and therefore, difficult to discover. Consequently, it is also difficult to gain sufficient assurances that a protocol is free of flaws. The approach of verifying the protocol for a few number of specific network topologies can never be exhaustive, and thus, it is far from being satisfactory as a method for security analysis. Hence, I advocate a more systematic approach to analysing the security of ad hoc and sensor network routing protocols, in which precise security definitions can be given and proofs can be carried out.

2.2 The Formal Framework of Wireless Routing Security

Before describing the details, I give a high level overview of my framework here. In my approach, a model is constructed for the protocol under investigation that is called the *dynamic model*. This model describes the operation of the protocol with all its details in a particular computational model. The model also contains an adversary that is an arbitrary process, which means that it may not follow the protocol rules faithfully, and it is only constrained to run in polynomial time. This allows us to consider any feasible attacks, which makes the model general. Instead of constructing an ideal-world model according to the standard simulation paradigm like in [Bellare *et al.*, 1998; Pfitzman and Waidner, 2001], I use a security objective function to represent the security objective of the protocol under investigation. In particular, this function is applied to the ensemble of the routing entries of honest nodes and decides whether the entries satisfy a certain security objective or not. The security objective also has to incorporate the tolerable imperfections of the system. Recall that the tolerable imperfections of the model are those attacks that are unavoidable, or they are too costly to defend against, and hence, we rather tolerate them. The protocol is said to be secure if the protocol executed in the dynamic model violates the security objective only with a negligible probability.

2.2.1 Adversary model

The adversary model is based on the informal description given in Section 2.1.4. According to this, I assume that the adversary can capture honest nodes in my model, and it may be able to compromise their cryptographic secrets (assuming that such secrets are used in the system). Thus, I assume in my model that the adversary can compromise cryptographic material (i.e., the adversary is an *insider adversary* in this sense). In addition, all adversarial nodes may be able to communicate in in-band (e.g., by tunneling) or out-of-band channels (e.g., other frequency channel or direct wired connection), which may be used to create wormholes. Therefore, it is also quite natural that all adversarial nodes can use all compromised cryptographic secrets. Finally, I assume that the initiator as well as the target of every route discovery process which are analysed in my model are always honest nodes. This is a minimal requirement which ensures that the initiator do not accept a route which is apparently forged (e.g., an adversarial initiator could accept a route which though contains incorrect signatures or Message Authentication Code (MAC)s). In order to be compliant with earlier security models (like in [Hu *et al.*, 2002]), the target is also assumed to be honest. I note that the

adversary is not restricted to initiate route discoveries, but we consider only those routing entries which are caused by route discoveries where the initiator and target are both honest nodes.

2.2.2 Network model

I assume that each honest device has exactly one transceiver in the network. If the adversary uses several transceivers I represent each of them by a distinct node. The network nodes are considered to be static (at least during the analysis), and I further assume that there is a single base station in sensor networks.

The honest nodes in the network are denoted by $v_0, \ldots, v_k$, where v_0 denotes the base station in case of sensor networks, and adversarial nodes are denoted by $v_{k+1}, \ldots, v_{k+m}$. The set of all nodes in the network is denoted by V, and the set of adversarial nodes is denoted by V^*, where $|V| = n = m + k + 1$, and $|V^*| = m$.

In order to model the connectivity between the nodes, I introduce a matrix $\mathbf{E}$, called *reachability matrix*, with size $n \times n$. Here, $E_{i,j}$ $(0 \leq i, j \leq n-1)$ represents the output power level which can be used by v_i and needed for v_i to communicate with v_j (i.e., if node v_i uses power level $E_{i,j}$, which is also denoted by e_{v_i,v_j}, to broadcast a message, then v_j also receives the message). In case v_j cannot receive any messages from v_i (e.g., there are obstacles between them, or the needed output power level is too high for v_i), $E_{i,j} = \infty$

I assume that each honest node can use a single globally unique identifier in the network, and these identifiers are authenticated in some way (e.g., by cryptographic means). I denote the set of these identifiers by L, and there is a function $\mathcal{L} : V \to 2^{L \cup \{\mathsf{undef}\}}$ that assigns a set of identifiers to each node, where $\mathsf{undef} \notin L$ is assigned to those nodes which have no identifiers in the network. According to my adversary model, I assume that the adversary has (authenticated) identifier(s) in the network, denoted by L^* that can be used by all adversarial nodes (i.e., $\mathcal{L}(v_{k+j}) = L^*$ for all $1 \leq j \leq m$). Moreover, for every honest node v_i $(0 \leq i \leq k)$, $\mathcal{L}(v_i)$ is a singleton, and $\mathcal{L}(v_i) \notin L^*$.

Finally, a *cost function* $\mathcal{C} : V \to \mathbb{R}$ assigns a (routing) cost value to each node in the network (e.g., the minimal processing delay, or constant 1 to each node in order to represent hop-count, etc.) that could influence the routing decisions[3].

Configuration: A *configuration* of a network is a quintuple $conf = (V, V^*, \mathcal{L}, \mathbf{E}, \mathcal{C})$, where V and V^* are the set of honest nodes and the set of adversarial nodes, resp., $\mathbf{E}$ is the reachability matrix, $\mathcal{L}$ is the labelling function, and $\mathcal{C}$ is the cost function.

I rely on the assumption that the configuration is static (at least during the time interval that is considered in the analysis). Thus, we view the route discovery part of routing protocols as a distributed algorithm that operates on this static configuration.

[3]Note that some routing protocols also incorporate link costs (e.g., packet loss ratio) into the calculation of a routing metric, however, the security of these protocols is not considered in this dissertation.

2.2.3 Security objective function

Diverse network applications entail different requirements for routing protocols. For instance, remote surveillance applications may require minimal delay for messages, while sensor applications performing some statistical measurements favour routing protocols prolonging network lifetime. The diversity of routing protocols is caused by these conflicting requirements: e.g., shortest-path routing algorithms cannot maximize the network lifetime, since always choosing the same nodes to forward messages causes these nodes to run out of their energy supply sooner. Several routing protocols use a trade-off to satisfy conflicting requirements [Singha *et al.*, 1998; Li *et al.*, 2001].

This small reasoning also points out that one cannot judge the utility of all routing protocols uniformly. Without a unified metric of utility we cannot refine our security objectives for routing protocols. For instance, according to the above example, a routing protocol that is secure against attacks aiming at decreasing network-lifetime cannot be secure against attacks aiming at increasing network delay. I model the negatively correlated requirements of routing, and essentially, our security objectives in a very general manner.

The state of the system is represented by the ensemble of the routing entries of *all* honest nodes in the network. The reason that I consider the result of the protocol with respect to the honest nodes exclusively is that the adversarial nodes may not follow the protocol rules faithfully. As the specification of routing entries depends on the routing protocol that is under investigation, the definition of system state is also protocol dependant. The security objective function $\mathcal{F} : \mathbb{G} \times \mathbb{S} \to \{0, 1\}$ is a binary function, where $\mathbb{S}$ denotes the set of all system states of all configurations, and $\mathbb{G}$ denotes the set of all configurations. Let $\mathcal{F}$ return 0 for all pairs of system states and configurations that are incorrect, otherwise it returns 1 (or vice-versa). This function intends to distinguish "attacked" (incorrect) states from "non-attacked" (correct) states. In the rest of the dissertation, I assume that $\mathcal{F}$ returns 0 for all incorrect states, otherwise it returns 1.

2.2.4 Dynamic model

The dynamic model is related to the simulation paradigm that has been successfully used to analyse the security of various cryptographic protocols so far [Bellare *et al.*, 1998; Shoup, 1999; Pfitzman and Waidner, 2001]. However, my model deviates from these works in the sense that I do not distinguish a real-world model and an ideal-world model as usual in the simulation paradigm, but I define a single model that represents the real operation of the network which contains an adversary. This adversary is not constrained apart from requiring it to run in polynomial time. This enables us to be concerned with arbitrary feasible attacks. The security objective function is applied to the output of this model (i.e., the final state of the system) in order to decide whether the protocol functions correctly or not. Once the model is defined, the goal is to prove that for any adversary, the probability that the security objective function is not satisfied is negligible.

The model that corresponds to a configuration $conf = (V, V^*, \mathcal{L}, \mathbf{E}, \mathcal{C})$ and adversary $\mathcal{A}$ is denoted by $sys_{conf,\mathcal{A}}$, and it is illustrated on Figure 2.7. I model the operation of the

protocol participants by interactive and probabilistic Turing machines, where the interaction is realized via common tapes. Correspondingly, I represent the adversary, the honest nodes, and the broadcast nature of the radio communication by machines A, M_i, and C, respectively. These machines communicate with each other via common tapes.

Each machine must be initialized with some input data (e.g., cryptographic keys, reachability matrix, etc.), which determines its initial state. Moreover, the machines are also provided with some random input (the coin flips to be used during the operation). Once the machines have been initialized, the computation begins. The machines operate in a reactive manner (i.e., they need to be activated in order to perform some computation). When a machine is activated, it reads the content of its input tapes, processes the received data, updates its internal state, writes some output on its output tapes, and goes back to sleep. The machines are activated in rounds by a hypothetic scheduler, and each machine in each round is activated only once. The order of activation is arbitrary with the only restriction that C must be activated at the end of the rounds.

Now, I present the machines in more details:

- **Machine** C. This machine is intended to model the radio communication. It has input tapes out_i and out_j^*, from which it reads messages written by M_i and A, resp. It also has output tapes in_i and in_j^*, on which it writes messages to M_i and A, resp. C is also initialized by matrix E at the beginning of the computation.

  Messages on tape out_i can have the format $(\ell_{sndr}, cont, e, dest)$, where $\ell_{sndr} \in L$ is the identifier of the sender, $cont$ is the message content, e is the output power level to be used to determine the range of transmission, and $dest$ is the identifier of the intended destination $dest \in L \cup \{*\}$, where $*$ indicates broadcast message.

  Messages on tape out_j^* can have the following formats:

  - $(\mathsf{MSG}, \ell_{sndr}, cont, e, dest)$: MSG message models a normal broadcast message sent by the adversary to machine C with sender identifier $\ell_{sndr} \in L$, message content $cont$, output power level e, and identifier of the intended destination $dest \in L \cup \{*\}$.
  - (JAM, e): Special JAM message, that is sent by the adversary to machine C, models the jamming capability of the adversary. When machine C receives a message JAM, it performs the requested jamming by deleting all messages in the indicated range e around the jamming node, which means that those deleted messages are not delivered to the nodes (including the jammer node itself) within the jamming range.
  - $(\mathsf{DEL}, \ell_{tar}, e)$: Special DEL message, that is sent by the adversary to machine C, models the modification capability of the adversary in wireless sensor networks. When receiving a message DEL with identifier $\ell_{tar} \in L$, machine C does not deliver any messages sent to node $v' \in V$, where $\mathcal{L}(v') = \ell_{tar}$, if v' is within the indicated range e, except the adversarial node itself that will receive the deleted messages. This models the sophisticated jamming technique in wireless sensor networks that I described in Subsection 2.1.4.

In a more formal way, when reading a message $msg^*_{in} = (\mathsf{MSG}, \ell_{sndr}, cont, e, dest)$ from out^*_j, C determines the nodes which receive the message by calculating the set of nodes $V_e \subseteq V$, such that for all $v' \in V_e$ $e_{v_{k+j},v'} \leq e$. Finally, C processes msg^*_{in} as follows.

1. if $dest \in L \cup \{*\}$, then C writes
   - $msg_{out} = (\ell_{sndr}, cont, dest)$ to the input tapes of machines corresponding to honest nodes in V_e
   - $msg^*_{out} = (\mathsf{MSG}, \ell_{sndr}, cont, dest)$ to the input tapes of machines corresponding to adversarial nodes in $V_e \setminus \{v_{k+j}\}$
2. otherwise C discards msg^*_{in}

When reading a message $msg^*_{in} = (\mathsf{JAM}, e)$ from out^*_j, C determines the set of nodes which receive the message by calculating $V_e \subseteq V$, such that for all $v' \in V_e$ $e_{v_{k+j},v'} \leq e$. Afterwards, C does not write any messages within the same round to the input tapes of machines corresponding to V_e.

When reading a message $msg^*_{in} = (\mathsf{DEL}, \ell_{tar}, e)$ from out^*_j, C determines the set of nodes which receive the message by calculating $V_e \subseteq V$, such that for all $v' \in V_e$ $e_{v_{k+j},v'} \leq e$. Finally, C processes msg^*_{in} as follows.

1. if there exists $v_x \in V_e$ $(1 \leq x \leq k)$, such that $\mathcal{L}(v_x) = \ell_{tar}$, then C does not write any messages within the same round from tape out_x to the input tapes of machines corresponding to $V_e \setminus \{v_{k+j}\}$
2. otherwise C discards msg^*_{in}

When reading a message $msg_{in} = (\ell_{sndr}, cont, e, dest)$ from out_i, C determines the set of nodes which receive the message by calculating $V_e \subseteq V$, such that for all $v' \in V_e$ $e_{v_i,v'} \leq e$. Finally, C processes msg_{in} as follows.

1. if $dest \in L \cup \{*\}$, then C writes
   - $msg_{out} = (\ell_{sndr}, cont, dest)$ to the input tapes of machines corresponding to honest nodes in $V_e \setminus \{v_i\}$
   - $msg^*_{out} = (\mathsf{MSG}, \ell_{sndr}, cont, dest)$ to the input tapes of machines corresponding to adversarial nodes in V_e
2. otherwise C discards msg_{in}

- **Machine** M_i. This machine models the operation of honest sensor nodes, and it corresponds to node v_i. It has input tape in_i and output tape out_i, which are shared with machine C. The format of input messages must be $(\ell_{sndr}, cont, dest)$, where $dest \in L \cup \{*\}$. The format of output messages must be $(\ell_{sndr}, cont, e, dest)$, where ℓ_{sndr} must be $\mathcal{L}(v_i)$, $dest \in L \cup \{*\}$, and e indicates the transmission range of the message for C. When this machine reaches one of its final states or there is a time-out during the computation process, it outputs its routing table (i.e., the set of all routing entries).

- **Machine** A. This machine models the adversary logic. Encapsulating each adversarial node into a single machine allows us to model wormholes inside A. One can imagine that the adversary deploys several transceivers in the network field, which are connected to a central adversary logic. In this convention, node v_{k+j} corresponds to an adversarial transceiver, which is modelled by input tape in_j^* and output tape out_j^*. These tapes are shared with machine C.

  The format of input messages must be $msg_{in}^* = (\mathsf{MSG}, \ell_{sndr}, cont, e, dest)$, where $dest \in L \cup \{*\}$.

  The format of output messages msg_{out}^* can be

  - $(\mathsf{MSG}, \ell_{sndr}, cont, e, dest)$, where $dest \in L \cup \{*\}$ and e indicates the transmission range of the message;
  - (JAM, e), where e indicates the range of jamming;
  - $(\mathsf{DEL}, \ell_{tar}, e)$, where e indicates the range of selective jamming, and $\ell_{tar} \in L$.

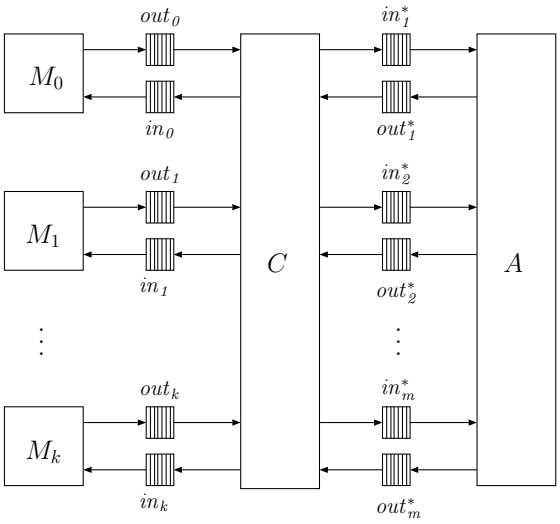


Figure 2.7: The dynamic model.

The computation ends, when all machines M_i reach their final states, or there is a time-out. The output of $sys_{conf,\mathcal{A}}$ is the value of the security objective function $\mathcal{F}$ applied to the resulted system state defined in Subsection 2.2.3 and configuration $conf$. The system state is represented by the ensemble of the routing entries of machines M_i. I denote the output by $Out_{conf,\mathcal{A}}^{\mathcal{F}}(r)$, where r is the random input of the model. In addition, $Out_{conf,\mathcal{A}}^{\mathcal{F}}$ will denote the random variable describing $Out_{conf,\mathcal{A}}^{\mathcal{F}}(r)$ when r is chosen uniformly at random.

2.2.5 Definition of secure routing

I denote the security parameter of the model by κ (e.g., κ is the key length of the cryptographic primitive employed in the routing protocol, such as MAC, digital signature etc.). Based on the model described in the previous subsections, I define routing security as follows:

Definition 1 *A routing protocol is secure with respect to security objective function $\mathcal{F}$, if for any configuration conf and any adversary $\mathcal{A}$, the probability that $Out^{\mathcal{F}}_{conf,\mathcal{A}}$ equals to zero is a negligible function of κ.*[4]

More intuitively, if a routing protocol is secure, then any system using this routing protocol may not satisfy its security objectives represented by function $\mathcal{F}$ only with a probability that is a negligible function of κ. This negligible probability is related to the fact that the adversary can always forge the cryptographic primitives (e.g., generate a valid MAC) with a very small probability depending on the value of κ.

2.2.6 Proof technique

In order to prove the security of a given routing protocol, one has to show that for any configuration *conf* and any adversary $\mathcal{A}$ the security objective function $\mathcal{F}$ returns 0 only with a probability that is a negligible function of the security parameter κ.

In particular, by proving the security of a protocol, we must show that those system states which violate our security objective (i.e., there is a configuration *conf* such that applying function $\mathcal{F}$ to those system states with *conf* results in 0) occur only with a negligible probability. However, even the number of all configurations for a given number of nodes is an exponential function of the number of all nodes. Thus, proving the security of a protocol by searching for all pairs of system states and configurations and test whether $\mathcal{F}$ returns 0 with these pairs seems to be a hard problem at first sight. However, as we will later see, all such pairs can be *reduced* to a few cases for all protocols which are analysed in this work. Then, we must prove that each of these cases occurs only with a negligible probability which concludes that the protocol satisfies Definition 1. In order to do this, we show that these cases can only occur in the model, if the adversary successfully breaks at least one cryptographic primitive (like the applied MAC, digital signature, or encryption scheme) used by the routing protocol. However, assuming that the applied primitives are secure, the probability of this event is a negligible function of the length of the security parameter (i.e., κ in my model).

In practice, failure of a proof usually indicates a problem with the protocol, and often, one can construct an attack by looking at where the proof failed.

2.2.7 Example: Insecurity of authenticated TinyOS beaconing

In this subsection, I present a routing state pollution attack against the secured TinyOS beaconing. The attack exploits the fact that, similar to SAODV in Subsection 4.1, routing messages are not authenticated between neighboring sensor nodes.

Operation of authenticated TinyOS beaconing

A lightweight cryptographic extension is employed in [Perrig *et al.*, 2002] in order to authenticate the beacon by the base station in TinyOS beaconing. This authenticated variant

[4] a function $\mu(x) : \mathbb{N} \rightarrow \mathbb{R}$ is negligible, if for every positive integer c and all sufficiently large x's (i.e., there exists an $N_c > 0$ for all $x > N_c$), $\mu(x) \leq x^{-c}$

of TinyOS routing uses μTimed, Efficient, Streaming, Loss-tolerant authentication protocol (Tesla) scheme to provide integrity for the beacon; each key is disclosed by the next beacon in the subsequent beaconing interval. In order to ease the demonstration of attacks that are described in the next subsection, I present a variant of this protocol which provides the "same" security as the authenticated routing protocol in [Perrig *et al.*, 2002]. Consequently, the presented attack against this new protocol also works against the protocol in [Perrig *et al.*, 2002]. Note that this protocol is only intended for demonstration rather than to be considered as a proposal of a new sensor routing protocol.

I assume that the base station B has a public-private key pair, where the public key is denoted by K_{pub}. Furthermore, it is assumed that each sensor node is also deployed with K_{pub}, and they are capable to perform digital signature verification with K_{pub}. Note that B never relays messages between sensor nodes.

Initially, B creates a beacon, that contains a constant message identifier BEACON, a randomly generated number rnd, the identifier of the base station Id_B, and a digital signature sig_B generated on the previous elements except Id_B. Afterwards, the base station floods the network by broadcasting this beacon:

$$B \rightarrow * \quad : \quad msg_1 = (\mathsf{BEACON}, \mathsf{rnd}, Id_B, \mathsf{sig}_B)$$

Each sensor node X receiving msg_1 checks whether it has already received a beacon with the same rnd in conjunction with a correct signature before. If it is true, the node discards msg_1, otherwise it verifies sig_B. If the verification is successful, then X sets Id_B as its parent, and re-broadcasts the beacon by changing the sender identifier Id_B to its own identifier Id_X:

$$X \rightarrow * \quad : \quad msg_2 = (\mathsf{BEACON}, \mathsf{rnd}, Id_X, \mathsf{sig}_B)$$

If the signature verification is unsuccessful, then X discards msg_1. Every sensor node receiving msg_2 performs the same steps what X has done before.

Optionally, B can initiate this topology construction periodically by broadcasting a new beacon with different rnd.

An attack against authenticated TinyOS beaconing

Before describing the security objective of authenticated TinyOS beaconing, I introduce the definition of *pseudo neighbors*. As the adversary can exchange messages between adversarial nodes through in-band (e.g., tunnelling) or out-of-band channels (e.g., wormholes) which is very costly to defend against, a pair of honest nodes can be "neighbors", if both of them have an adversarial neighbor.

Definition 2 (Pseudo neighbors) *Two honest nodes $v_i, v_j \in V \setminus V^*$ $(i \neq j)$ are pseudo neighbors, if and only if there exist x, y $(k+1 \leq x, y \leq k+m)$ such that $E_{i,x} = 1$ and $E_{j,y} = 1$, and $v_x, v_y \in V^*$.*

Two nodes are pseudo neighbors, only if each of them has an adversarial neighbor. In the sequel, we distinguish pseudo neighbors from direct neighbors; two honest nodes v_i, v_j are

direct neighbors, if $E_{i,j} = 1$. However, note that being direct neighbors and pseudo neighbors are not exclusive.

I recall that the state of the system is the ensemble of the routing entries of all *honest* nodes. This system state with a given configuration *conf* is represented by a matrix T^{conf} with size $(k+1) \times (k+2)$ in case of authenticated TinyOS beaconing, and I refer to this as a *routing topology* with configuration *conf* in the sequel. For all $0 \leq i \leq k$,

- and $0 \leq j \leq k$, $T^{conf}_{i,j} = 1$, if honest node v_i sends every message to an honest node v_j in order to deliver the message to the destination node, otherwise let $T^{conf}_{i,j}$ be 0.
- $T^{conf}_{i,k+1} = 1$, if honest node v_i sends every message to an adversarial node in order to deliver the message to the destination node, otherwise let $T^{conf}_{i,j}$ be 0.

Note that the rows and columns of T^{conf} are numbered from zero, and the same holds for all matrices in the rest of the dissertation.

In this way, a routing topology can also be considered as a directed graph described by matrix T^{conf}. In fact, T^{conf} is a random variable, where the randomness is caused by the sensor readings initiated randomly by the environment, processing and transmission time of the sensed data, etc. In the sequel, I will omit the index *conf* of T^{conf} when the configuration can be unambiguously determined in a given context.

I recall that the set of all configurations is denoted by $\mathbb{G}$. Therefore, the security objective function $\mathcal{F} : \mathbb{G} \times \mathbb{S} \to \mathbb{R}$ assigns a real number to a random routing topology of a configuration. This function intends to distinguish "attacked" topologies from "non-attacked" topologies based on a well-defined security objective. For example, let us consider routing protocols that build a routing tree, where the root is the base station. We can compare routing trees based on network lifetime by the following security objective function

$$\mathcal{F}(conf, T^{conf}) = \begin{cases} 1, & \frac{1}{k}\sum_{i=1}^{k} \mathcal{E}(v_i, conf, T^{conf}) \leq t_e \\ 0, & \text{otherwise} \end{cases}$$

where $\mathcal{E} : V \times \mathbb{G} \times \mathbb{S} \to \mathbb{R}$ assigns the overall energy consumption of the path from a node v_i to v_0 (the base station) in a routing tree of a configuration, and t_e is a predefined threshold. Since T^{conf} is a random variable, the output of $\mathcal{F}$ is a random variable too. If the distribution of this output non-negligibly differs from 1 (i.e., the overall energy consumption non-negligibly exceeds t_e), then the protocol is not secure. If we intend to compare routing trees based on network delay a simple security objective function may be

$$\mathcal{F}(conf, T^{conf}) = \begin{cases} 1, & \frac{1}{k}\sum_{i=1}^{k} \mathcal{M}(v_i, conf, T^{conf}) \leq t_d \\ 0, & \text{otherwise} \end{cases}$$

where $\mathcal{M} : V \times \mathbb{G} \times \mathbb{S} \to \mathbb{R}$ assigns the length of the path from a node to v_0 in a routing topology of a configuration, and t_d is a predefined threshold.

A simple security objective of authenticated TinyOS beaconing can be to guarantee the correctness of all routing entries in the network. Namely, it is desirable that a sender node v_i

is always able to reach node v_j, if v_i set $\mathcal{L}(v_j)$ as its parent identifier earlier. It means that if node v_i sets node $\mathcal{L}(v_j)$ as its parent identifier, then $E_{i,j}$ should contain a finite value, or v_i as well as v_j should be direct or pseudo neighbors.

Let the security objective function of authenticated TinyOS beaconing, denoted by $\mathcal{F}^{ATB}$, return 1 for all pairs of system states and configurations where for all i, j, if $T_{i,j} = 1$, then v_i and v_j are direct or pseudo neighbors. Otherwise, $\mathcal{F}$ returns 0.

Theorem 1 *Authenticated TinyOS beaconing is insecure with respect to* $\mathcal{F}^{ATB}$.

Proof I will show that authenticated TinyOS beaconing is not secure in my model with respect to security objective function $\mathcal{F}^{\mathrm{ATB}}$. In particular, I present a configuration $conf'$ and an adversary $\mathcal{A}$, for which there exists i, j such that $T_{i,j} = 1$, but v_i and v_j are neither pseudo nor direct neighbors. Moreover, the success probability of the adversary $\mathcal{A}$ described below is independent from κ.

In the sequel, I will refer to non-adversarial machines with their identifiers. The configuration $conf'$ and the result of the attack is illustrated in Figure 2.8. I assume that the base station broadcasts only a single beacon during the analysis (i.e., only a single beaconing interval is considered).

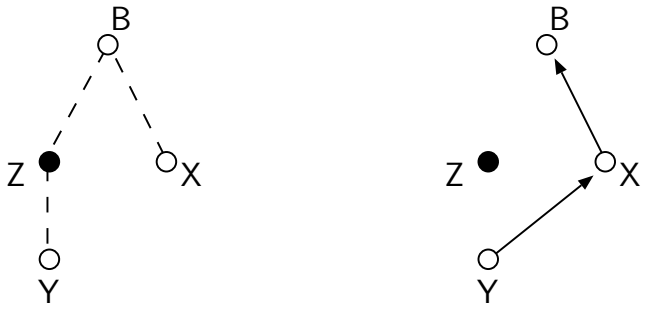


Figure 2.8: A simple attack against authenticated TinyOS beaconing. B, X, and Y are honest nodes, whereas Z is an adversarial node. Links are assumed to be bidirectional. The network topology is illustrated on the left-hand side, where a dashed line denotes a direct link. The resulted routing topology is depicted on right-hand side, where Y falsely sets X as its parent identifier, however, there is no wireless link between them, even more there is no wireless link between X and Z.

At the beginning, the base station B floods the network by a beacon

$$B \rightarrow * \quad : \quad msg'_1 = (\mathsf{BEACON}, \mathsf{rnd}, B, \mathsf{sig}_B)$$

The adversarial machine (node Z) and honest node X receive this beacon, and X sets B as its parent, since the verification of the signature is successful. X modifies the beacon by replacing sender identifier B to X, and broadcasts the resulted beacon:

$$X \rightarrow * \quad : \quad msg'_2 = (\mathsf{BEACON}, \mathsf{rnd}, X, \mathsf{sig}_B)$$

In parallel, the adversarial machine modifies the beacon by replacing sender identifier B to X, and broadcasts the resulted beacon:

$$Z \rightarrow * \quad : \quad msg'_2 = (\mathsf{BEACON}, \mathsf{rnd}, X, \mathsf{sig}_B)$$

Upon the reception of msg'_2, node Y sets X as its parent, since sig_B is correct. However, B and Y are neither pseudo nor direct neighbors. ■

Note that the above attack may not work, if the nodes execute a neighbor discovery protocol. In that case, Y can detect that X does not belong to Y's neighborhood. However, the attack described in Subsection 2.1.6 against TinyOS beaconing also works against the authenticated version of TinyOS beaconing.

2.3 Summary

In this chapter, I first informally defined the problem of secure routing in multi-hop wireless networks. Then, I described the objectives of a malicious user (the adversary) who wants to subvert the normal operation of routing protocols in ad hoc and sensor networks. Afterwards, I defined the capabilities of such an adversary, and I investigated the deviations of this adversary model from the standard adversary model assumed so far in traditional wired networks. After that, I gave an overview of the fundamental attack methods employed by the adversary against ad hoc and sensor network routing protocols. I also introduced routing state pollution attacks which are considered in this dissertation. In the following chapters, I will show that even if the adversary uses the basic message manipulation techniques described in Section 2.1.4, it is able to successfully mount routing state pollution attacks against well-known "secure" ad hoc and sensor network routing protocols like Ariadne [Hu *et al.*, 2002], or SAODV [Zapata and Asokan, 2002]. The formal framework presented in the second part of this chapter will be used to analyse the security of these protocols.

Chapter 3

Secure Dynamic Source Routing in Wireless Ad hoc Networks

In this chapter, I demonstrate how my model described in Chapter 2 can be used to analyse the security of dynamic source routing protocols in wireless ad hoc networks. I define a general security objective of dynamic source routing, and I show that Ariadne with digital signatures is insecure considering this security objective. This motivates the development of a novel secure source routing protocol in this chapter called endairA which is, in turn, provably secure in my model regarding the same security objective. This also proves that my model is capable of distinguishing between source routing protocols in terms of security.

The attacks presented in Subsection 2.1.6 clearly show that security flaws in ad hoc and sensor network routing protocols can be very subtle. In this chapter, I describe a routing state pollution attack against Ariadne which causes an honest node to accept such routes that do not exist in the underlying network topology. Even more, in order to mount this attack, the adversary only needs a single adversarial node in the network, which increases the severity of this vulnerability. Thus, inspired by Ariadne, I design a new source routing protocol for wireless ad hoc networks. I call the protocol endairA (which is the reverse of Ariadne), because instead of signing the route request, I propose that intermediate nodes should sign the route reply. I also demonstrate the usefulness of my model by showing that endairA is secure in my model, whereas Ariadne is not.

The organisation of this chapter is as follows. First, in Section 3.1, I present the operation of a secure source routing protocol called Ariadne. Then, in Section 3.2, I define the general security objective of dynamic source routing in wireless ad hoc networks. The tolerable imperfections of the model, which are incorporated by the security objective, are detailed in Section 3.3. Then, in Section 3.4, I show that Ariadne is insecure considering this security objective by presenting a routing state pollution attack against that. In Section 3.5, I present the security analysis of a new secure source routing protocol, called endairA. In particular, in Subsection 3.5.1, I detail the network and node assumptions of endairA. Then, in Subsection 3.5.2, I specify the basic version of endairA by presenting its operation. This is followed by

the security proof of endairA in Subsection 3.5.3. I discuss possible extensions and variants of endairA in Subsection 3.5.4. Finally, in Section 3.6, I conclude this chapter.

3.1 Operation of the basic Ariadne protocol with digital signatures

Ariadne has been proposed in [Hu *et al.*, 2002] as a secure on-demand source routing protocol for ad hoc networks. Ariadne comes in three different flavours corresponding to three different techniques for data authentication. More specifically, authentication of routing messages in Ariadne can be based on Tesla [Perrig *et al.*, 2000], on digital signatures, or on MACs. We discuss Ariadne with digital signatures.

The initiator of the route discovery generates a route request message and broadcasts it to its neighbors. The route discovery message contains the identifiers of the initiator and the target, a randomly generated request identifier, and a MAC computed over these elements with a key shared by the initiator and the target. This MAC is hashed iteratively by each intermediate node together with its own identifier using a publicly known one-way hash function. The hash values computed in this way are called per-hop hash values. Each intermediate node that receives the request for the first time re-computes the per-hop hash value, appends its identifier to the list of identifiers accumulated in the request, and generates a digital signature on the updated request. Finally, the signature is appended to a signature list in the request, and the request is re-broadcast. When the target receives the request, it verifies the per-hop hash by re-computing the initiator's MAC and the per-hop hash value of each intermediate node. Then it verifies all the digital signatures in the request. If all these verifications are successful, then the target generates a route reply and sends it back to the initiator via the reverse of the route obtained from the route request. The route reply contains the identifiers of the target and the initiator, the route and the list of digital signatures obtained from the request, and the digital signature of the target on all these elements. Each intermediate node passes the reply to the next node on the route (towards the initiator) without any modifications. When the initiator receives the reply, it verifies the digital signature of the target and the digital signatures of the intermediate nodes (for this it needs to reconstruct the requests that the intermediate nodes signed). If the verifications are successful, then it accepts the route returned in the reply.

Although Ariadne does not specify it explicitly, we will nonetheless assume that each node also performs the following verifications when processing route request and route reply messages:

- When a node v receives a route request for the first time, it verifies if the last identifier of the accumulated route in the request corresponds to a neighbor of v. If no identifiers can be found in the accumulated route, then v verifies if the identifier of the initiator corresponds to a neighboring node.
- When a node v receives a route reply, it verifies if its identifier is included in the route carried by the reply. In addition, it also verifies if the preceding identifier (or if there is

no preceding identifier, then the identifier of the initiator) and the following identifier (or if there is no following identifier, then the identifier of the target) in the route correspond to neighbors of v.

If these verifications fail, then the message is dropped.

3.2 Security objective

Intuitively, the minimum that one may require from the route discovery part of the routing protocol is that it returns only existing routes. My security objective is based on this intuition. It is clear that the security of routing may be viewed more broadly, including other issues such as detecting and avoiding nodes that drop data packets, but I recall that my model is only intended for modelling routing state pollution attacks. I deliberately restrict myself to this small subset of all possible attacks, because it is already challenging to properly formalize that.

Taking into account that the adversary can exchange messages between adversarial nodes through in-band (e.g., tunnelling) or out-of-band channels (e.g., wormholes) which is very costly to defend against, I introduce the definition of *plausible routes*.

Definition 3 (Plausible route) *A sequence $\ell_1, \ell_2, \ldots, \ell_n$ of identifiers is a plausible route with respect to configuration conf, if each of the identifiers $\ell_1, \ell_2, \ldots, \ell_n$ is different and there exists a sequence $v_1, v_2, \ldots, v_t$ $(2 \leq t \leq n)$ of honest nodes such that*

- *$\mathcal{L}(v_1) = \ell_1$ and $\mathcal{L}(v_t) = \ell_n$;*
- *for all $1 \leq i \leq t-1$,*
  - *there exists $1 \leq j, d \leq n-1$ such that $\mathcal{L}(v_i) = \ell_j$ and $\mathcal{L}(v_{i+1}) = \ell_{j+d}$, where for all $j+1 \leq z \leq j+d-1$, $\ell_z \in L^*$;*
  - *v_i and v_{i+1} are direct or pseudo neighbors.*

Examples for plausible routes are depicted in Figure 3.1.

I recall that the security objective function $\mathcal{F} : \mathbb{G} \times \mathbb{S} \rightarrow \{0, 1\}$ is a binary function, where $\mathbb{S}$ denotes the set of all system states of all configurations, and $\mathbb{G}$ denotes the set of all configurations. I also recall that a system state represents the set of all routing entries of all *honest* nodes in the network. In the case of source routing, a routing entry includes a route (i.e., a sequence of node identifiers) that is used for data forwarding towards the destination, which is the last element of this route. Let $\mathcal{F}$ of secure dynamic source routing return 0 for all pairs of system states and configurations where the system state contains a non-plausible routes with respect to the configuration, where this non-plausible route belongs to an honest node. Otherwise, $\mathcal{F}$ returns 1.

According to Definition 1, if any honest node in the model returns a non-plausible route with non-negligible probability for a given configuration, then the protocol is insecure. In other words, a dynamic source routing protocol is secure, if it returns non-plausible route only with a probability that is a negligible function of κ.

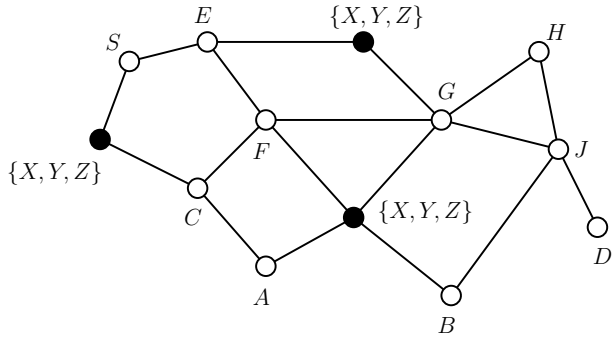


Figure 3.1: Examples for plausible routes. Adversarial nodes are denoted by black-filled circles, and the corrupted identifiers are X, Y, and Z. According to Definition 3, (S, E, F, X, Z, B, J, D) as well as (S, E, Y, Z, X, B, J, D) are plausible routes. However, (S, Y, C, A, Y, B, J, D) is not a plausible route due to the repeating identifier Y. Similarly, (S, E, Y, H, J, D) is not a plausible route either, because none of the adversarial nodes have any neighbors with identifier H.

3.3 Tolerable imperfections

In this section, I explain the tolerable imperfections of my model in more details. Recall that those attacks are considered to be the tolerable imperfections that are unavoidable or too costly to defend against, and thus, we rather tolerate them. In other words, a routing protocol that is secure in my model may not be resistant to these types of attacks. Most of these attacks are built on the delay and deletion of messages, and the in-band as well as the out-of-band channel attacks.

The rationale behind the definition of pseudo neighbors and plausible routes in Definitions 2 and 3, resp., is that two adversarial nodes, who may be located on different network parts, are able to transfer the signatures of honest nodes by using out-of-band channels like wormholes or some in-band channels (assuming that these honest nodes believe that they are neighbors). In the latter case, signatures are transferred as a part of an existing message to remote adversarial nodes. For instance, one adversarial node captures the signature of an honest neighbor denoted by H, then fragments the signature, and puts these fragments into new RREQ messages as their random identifiers. Finally, the adversarial node sends these fabricated messages to a remote adversarial node by initiating new route discoveries towards that node. As these random identifiers are indistinguishable from signatures (both are random values for a given request message), these fabricated messages will pass all verifications at intermediate honest nodes and will reach the remote adversarial node. When this remote adversarial node receives all fragments, it can successfully impersonate H by reconstructing the signatures from the fragments. In this case, the adversary uses a side-channel provided by the protocol messages to impersonate honest nodes, and thus these attacks are also called as side-channel attacks [Burmester and de Medeiros, 2009]. Moreover, when the adversary can add a corrupted identifier to a route, it can also add as many corrupted identifier as it has, since it can authenticate each of them.

The reason that I tolerate in-band and out-of-band attacks is twofold. First, for most real scenarios side-channel attacks are impractical for the adversary, as by the time the last fragment is successfully transferred, the request or reply message becomes obsolete. Second, these attacks can be mitigated but, to the best of my knowledge, they are not avoidable completely. Therefore, I consider these attacks as some of the tolerable imperfections of my model.

Finally, I note that the authors in [Kim and Tsudik, 2005] also identified some of these unavoidable attacks caused by colluding adversarial nodes.

3.4 Insecurity of Ariadne

In [Buttyán and Vajda, 2004], the authors discovered a new routing state pollution attack against Ariadne. In this section, I describe this attack in my model.

Theorem 2 *Ariadne is an insecure source routing protocol for wireless ad hoc networks.*

Proof I show that there exists a configuration $conf'$ and an adversary $\mathcal{A}$ such that a route reply message in $sys_{conf',\mathcal{A}}$ causes an honest node to accept a non-plausible route with non-negligible probability.

In what follows, I will refer to non-adversarial machines with their identifiers. Let us consider Figure 3.2, which illustrates part of a $conf'$. The single adversarial node, and thus the adversarial machine, is denoted by A. Let us assume that S sends a route request towards D. The request reaches V that rebroadcasts it. Thus, A receives the following route request message:

$$msg_1 = (\mathsf{rreq}, \mathsf{S}, \mathsf{D}, id, h_{\mathsf{V}}, (\ldots, \mathsf{V}), (\ldots, sig_{\mathsf{V}}))$$

where id is the random request identifier, h_{V} is the per-hop hash value generated by V, and sig_{V} is the signature of V. A does not re-broadcast msg_1. Later, A receives another route request from X

$$msg_2 = (\mathsf{rreq}, \mathsf{S}, \mathsf{D}, id, h_{\mathsf{X}}, (\ldots, \mathsf{V}, \mathsf{W}, \mathsf{X}), (\ldots, sig_{\mathsf{V}}, sig_{\mathsf{W}}, sig_{\mathsf{X}}))$$

From msg_2, A knows that W is a neighbor of V. A computes $h_{\mathsf{A}} = H(A, H(\mathsf{W}, h_{\mathsf{V}}))$, where h_{V} is obtained from msg_1, and H is the publicly known hash function used in the protocol. A obtains the signatures $\ldots, sig_{\mathsf{V}}, sig_{\mathsf{W}}$ from msg_2. Then, A generates and broadcasts the following request:

$$msg_3 = (\mathsf{rreq}, \mathsf{S}, \mathsf{D}, id, h_A, (\ldots, \mathsf{V}, \mathsf{W}, A), (\ldots, sig_{\mathsf{V}}, sig_{\mathsf{W}}, sig_A))$$

Later, D generates the following route reply and sends it back towards S:

$$msg_4 = (\mathsf{rrep}, \mathsf{D}, \mathsf{S}, (\ldots, \mathsf{V}, \mathsf{W}, A, \ldots), (\ldots, sig_{\mathsf{V}}, sig_{\mathsf{W}}, sig_A, \ldots), sig_{\mathsf{D}})$$

When A receives this route reply, it forwards it to V in the name of W. Finally, S will output

the route $(\mathsf{S}, \ldots, \mathsf{V}, \mathsf{W}, A, \ldots, \mathsf{D})$, which is a non-plausible route, as honest node W does not have any adversarial neighbor according to $conf'$. ■

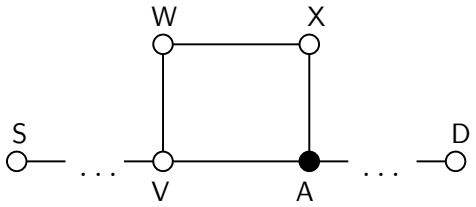


Figure 3.2: An attack against Ariadne. The single adversarial node is denoted by A, whereas honest nodes are denoted by S, V, W, X, D, resp. The source S initiates a route discovery towards destination D.

Note that the success probability of the above attack is independent from the security of the applied digital signature.

3.5 endairA: a provably secure source routing protocol

3.5.1 Assumptions

I assume that not all nodes are willing to forward packets for other nodes. In particular, some nodes, called adversarial nodes, behave as it is described in Subsection 2.1.4.

I assume bidirectional links in the network (i.e., if node A is able to transmit to some node B, then B is also able to transmit to A). If unidirectional links can occur, a mechanism is needed to eliminate such links. Furthermore, I disregard all attacks on the Medium Access Control protocol (like in [Bharghavan *et al.*, 1994; Committee, 1997]) which is in use. I assume that links are unreliable (i.e., packets may be lost, corrupted, re-ordered, or duplicated in transmission).

Ad hoc network nodes show a high variety in terms of computational capabilities. A typical node ranges from devices with little computational power such as PDAs to resource-rich nodes like laptops. endairA supports all such devices, it only requires each node to be capable of generating and verifying digital signatures like Digital Signature Algorithm (DSA) [dsa, 1994], Rivest Shamir Adleman asymmetric-key based cryptographic algorithms (RSA) [Rivest *et al.*, 1978] or Elliptic Curve Digital Signature Algorithm (ECDSA) [X9.63, 1999]. Nodes with less computational resources can use an optimized version of ECDSA. Some public key distribution mechanism is also assumed. This can be the simple pre-deployment of all necessary public keys into each node, or the employment of some Public Key Infrastructure (PKI), where the trusted Certification Authority's public key is pre-deployed into each node that is used to authenticate the public keys of other nodes. More sophisticated techniques are proposed in [Hubaux *et al.*, 2001]. I also assume that nodes can identify their neighborhood by executing a (secure) neighbor discovery protocol.

3.5.2 Specification of the basic endairA protocol

The operation and the messages of endairA are illustrated in Figure 3.3. In endairA, the initiator of the route discovery process generates a route request, which contains the identifiers of the initiator and the target, and a randomly generated request identifier. Each intermediate node that receives the request for the first time appends its identifier to the route accumulated so far in the request, and re-broadcasts the request. When the request arrives to the target, it generates a route reply. The route reply contains the identifiers of the initiator and the target, the accumulated route obtained from the request, and a digital signature of the target on these elements. The reply is sent back to the initiator on the reverse of the route found in the request. Each intermediate node that receives the reply verifies that its identifier is in the node list carried by the reply, and that the preceding identifier (or that of the initiator if there is no preceding identifier in the node list) and the following identifier (or that of the target if there is no following identifier in the node list) belong to neighboring nodes. Each intermediate node also verifies that the digital signatures in the reply are valid and that they correspond to the following identifiers in the node list and to the target. If these verifications fail, then the reply is dropped. Otherwise, it is signed by the intermediate node, and passed to the next node on the route (towards the initiator). When the initiator receives the route reply, it verifies if the first identifier in the route carried by the reply belongs to a neighbor. If so, then it verifies all the signatures in the reply. If all these verifications are successful, then the initiator accepts the route.

$S \rightarrow *$	:	$(\mathsf{rreq},\ S,\ T,\ id,\ ())$
$A \rightarrow *$	:	$(\mathsf{rreq},\ S,\ T,\ id,\ (A))$
$B \rightarrow *$	:	$(\mathsf{rreq},\ S,\ T,\ id,\ (A, B))$
$T \rightarrow B$	:	$(\mathsf{rrep},\ S,\ T,\ (A, B),\ (sig_T))$
$B \rightarrow A$	:	$(\mathsf{rrep},\ S,\ T,\ (A, B),\ (sig_T, sig_B))$
$A \rightarrow S$	:	$(\mathsf{rrep},\ S,\ T,\ (A, B),\ (sig_T, sig_B, sig_A))$

Figure 3.3: An example for the operation and messages of endairA. The initiator of the route discovery is S, the target is T, and the intermediate nodes are A and B. id is a randomly generated request identifier. sig_A, sig_B, and sig_T are digital signatures of A, B, and T, respectively. Each signature is computed over the message fields (including the signatures) that precede the signature.

I note that another secure source routing protocol called as Secure Route Discovery Protocol (SRDP) [Kim and Tsudik, 2005] applies the same principle as endairA. Namely, instead of signing the requests, each intermediate node only signs the reply messages. The focus of [Kim and Tsudik, 2005] is to explore different cryptographic techniques with different levels of security, efficiency, and robustness. In particular, they investigate aggregated MACs and several multi-signature schemes to authenticate reply messages. The common characteristic of these primitives is that the signature list in the route reply can be replaced with a single aggregate signature or MAC computed iteratively by the intermediate nodes in order to reduce communication overhead. By contrast, my work is focused on the design and formal

validation of secure source routing protocols. Finally, I note that we first published our results in [Ács *et al.*, 2005b], which was earlier than [Kim and Tsudik, 2005].

3.5.3 Security proof

The proof of the following theorem illustrates how the framework introduced in Section 2 can be used in practice.

Theorem 3 *endairA is a secure source routing protocol for wireless ad hoc networks, if the signature scheme is secure against chosen message attacks.*

Proof We want to show that for any configuration *conf* and any adversary $\mathcal{A}$, a route reply message in $sys_{conf,\mathcal{A}}$ causes any honest node to accept a non-plausible route only with negligible probability.

Similar to the proof of Theorem 2, I will refer to non-adversarial machines with their identifiers. Let us suppose that the following route reply is received by a non-adversarial machine ℓ_{ini} in $sys_{conf,\mathcal{A}}$:

$$msg = (\mathsf{rrep},\ \ell_{ini},\ \ell_{tar},\ (\ell_1, \ldots, \ell_p),\ (sig_{\ell_{tar}}, sig_{\ell_p}, \ldots, sig_{\ell_1}))$$

Let us suppose that *msg* passes all the verifications required by endairA at ℓ_{ini}, which means that all signatures in *msg* are correct, and ℓ_{ini} has a direct or pseudo neighbor that uses the identifier ℓ_1. Let us further suppose that *msg* has been received with a route $(\ell_{ini}, \ell_1, \ldots, \ell_p, \ell_{tar})$ which is non-plausible in *conf*. Hence, *msg* causes ℓ_{ini} to accept a non-plausible route.

It is easy to see that all sequences of identifiers can unambiguously be partitioned such that each non-compromised identifier form a single partition and all consecutive compromised identifiers (between two non-compromised identifiers) also form a single partition. Let $P_1, P_2, \ldots, P_k$ be such a unique partitioning of the route $(\ell_{ini}, \ell_1, \ldots, \ell_p, \ell_{tar})$. As $(\ell_{ini}, \ell_1, \ldots, \ell_p, \ell_{tar})$ is a non-plausible route, according to Definition 3, at least one of the following two statements holds:

- *Case 1:* There exist two partitions $P_i = \{\ell_j\}$ and $P_{i+1} = \{\ell_{j+1}\}$ such that both ℓ_j and ℓ_{j+1} are non-compromised identifiers, and there does not exist honest nodes v_x, v_y $(v_x, v_y \in V \setminus V^*)$ such that $\mathcal{L}(v_x) = \ell_j$ and $\mathcal{L}(v_y) = \ell_{j+1}$ and v_x, v_y are direct or pseudo neighbors.

- *Case 2:* There exist three partitions $P_i = \{\ell_j\}$, $P_{i+1} = \{\ell_{j+1}, \ldots, \ell_{j+q}\}$, and $P_{i+2} = \{\ell_{j+q+1}\}$ such that ℓ_j and ℓ_{j+q+1} are non-compromised and $\ell_{j+1}, \ldots, \ell_{j+q}$ are compromised identifiers, and there does not exist honest nodes v_x, v_y $(v_x, v_y \in V \setminus V^*)$ such that $\mathcal{L}(v_x) = \ell_j$ and $\mathcal{L}(v_y) = \ell_{j+q+1}$ and v_x, v_y are direct or pseudo neighbors.

I show that in both cases, the adversary must have forged the digital signature of a non-adversarial machine.

In Case 1, machine ℓ_{j+1} does not sign the route reply, since it is non-adversarial and it detects that the identifier that precedes its own identifier in the route does not belong to a neighboring machine. Hence, the adversary must have forged $sig_{\ell_{j+1}}$ in msg.

In Case 2, the situation is more complicated. Let us assume that the adversary has not forged the signature of any of the non-adversarial machines. Machine ℓ_j must have received

$$msg' = (\mathsf{rrep},\ \ell_{ini},\ \ell_{tar},\ (\ell_1, \ldots, \ell_p),\ (sig_{\ell_{tar}}, sig_{\ell_p}, \ldots, sig_{\ell_{j+1}}))$$

from an adversarial neighbor, say A, since ℓ_{j+1} is compromised, and thus, a non-adversarial machine would not send out a route reply message with $sig_{\ell_{j+1}}$. Therefore, ℓ_j has an adversarial neighbor. In order to generate msg', machine A must have received

$$msg'' = (\mathsf{rrep},\ \ell_{ini},\ \ell_{tar},\ (\ell_1, \ldots, \ell_p),\ (sig_{\ell_{tar}}, sig_{\ell_p}, \ldots, sig_{\ell_{j+q+1}}))$$

only from a non-adversarial machine because, by assumption, the adversary has not forged the signature of ℓ_{j+q+1}, which is non-compromised. However, the only non-adversarial machine that would send out msg'' is ℓ_{j+q+1}. This would mean that both ℓ_j and ℓ_{j+q+1} have an adversarial neighbor which means that they are pseudo neighbors. However, this contradicts the assumption of Case 2. This means that our original assumption cannot be true, and hence, the adversary must have forged the signature of a non-adversarial machine.

It should be intuitively clear that if the signature scheme is secure, then the adversary can forge a signature only with a probability that is a negligible function of κ, and thus, a route reply message in $sys_{conf,\mathcal{A}}$ causes an honest machine to accept a non-plausible route only with negligible probability. Nevertheless, I sketch how this could be proven formally. The proof is indirect. I assume that there exist a configuration $conf$ and an adversary $\mathcal{A}$ such that a route reply message in $sys_{conf,\mathcal{A}}$ causes an honest machine to accept a non-plausible route with probability ϵ, and then, based on that, we construct a forger F that can break the signature scheme with probability ϵ/n. If ϵ is non-negligible, then so is ϵ/n, and thus, the existence of F contradicts with the assumption about the security of the signature scheme.

The construction of F is the following. Let puk be an arbitrary public key of the signature scheme. Let us assume that the corresponding private key prk is not known to F, but F has access to a signing oracle that produces signatures on submitted messages using prk. F runs a simulation of $sys_{conf,\mathcal{A}}$ where all machines are initialized as described in the model, except that the public key of a randomly selected non-adversarial machine ℓ_i is replaced with puk. During the simulation, whenever ℓ_i signs a message m, F submits m to the oracle, and replaces the signature of ℓ_i on m with the one produced by the oracle. This signature verifies correctly on other machines later, since the public verification key of ℓ_i is replaced with puk. By assumption, with probability ϵ, the simulation of $sys_{conf,\mathcal{A}}$ will result in a route reply message msg such that all signatures in msg are correct and msg contains a non-plausible route. As we saw above, this means that there exists a non-adversarial machine ℓ_j such that msg contains the signature sig_{ℓ_j} of ℓ_j, but ℓ_j has never signed (the corresponding part of) msg. Let us assume that $i = j$. In this case, sig_{ℓ_j} is a signature that verifies correctly with

the public key puk. Since ℓ_j did not sign (the corresponding part of) msg, F did not call the oracle to generate sig_{ℓ_j}. This means that F managed to produce a signature on a message that verifies correctly with puk. Since F selected ℓ_i randomly, the probability of $i = j$ is $\frac{1}{n}$, and hence, the success probability of F is ϵ/n. ■

3.5.4 Practical extensions to the basic endairA protocol

Note that in my model presented in Section 2, I made the assumption that the nodes are static (at least during the period of time that is analyzed). The proof of security of endairA relies on this assumption. More precisely, in the proof, I show that if a route is returned by endairA to an honest node, then that route must be plausible with overwhelming probability. Moreover, once a route has been returned, it remains valid forever, because the graph does not change. This means that under the assumption of static nodes, the basic endairA protocol is not vulnerable to replay attacks. However, if we relax this assumption, and we allow the nodes to move, then the basic protocol has a problem. In that case, when a node initiates a route discovery process and the adversary receives a route request, it can replay an old route reply, and if that reply reaches the initiator, then it will be accepted, despite the fact that it may contain outdated information (i.e., a route that does not exist any more due to the mobility of the nodes, and thus it may not be plausible either).

Fortunately, we can easily extend the basic endairA protocol to mitigate this problem. All we need to do is to require the target of the route discovery to insert the random request identifier id (received in the route request) in the route reply. Hence, in the extended endairA protocol, the route reply that is passed from intermediate node F_i to node F_{i-1} looks as follows:

$$(\mathsf{rrep},\ S,\ T,\ id,\ (F_1, \ldots, F_n),\ (sig_T, sig_{F_n}, \ldots, sig_{F_i}))$$

Now, when the initiator receives a route reply, it also verifies if it received back the request identifier that it sent in the route request. This makes it practically impossible for the adversary to successfully replay an old route reply that belongs to a previous route discovery process. Of course, when nodes are allowed to move, it is possible that a route reply contains a non-existent route even if there was no attack at all. In order to alleviate this problem, the time interval within which the initiator accepts a reply with a specific request identifier should be appropriately limited.

Another problem with the basic endairA protocol is that it is vulnerable to malicious route request flooding attacks. This is because the route request messages are not authenticated in any way, and hence, an adversary (even without compromising any identity) can initiate route discovery processes in the name of honest nodes. These forged route discovery processes will be carried out completely, including the flooding of the route requests in the whole network, because only the impersonated initiators can detect that they are forged. In order to prevent this, the route request can be digitally signed by the initiator, and rate limiting techniques similar to the one used for Ariadne [Hu *et al.*, 2002] can be applied with endairA too. Naturally, such extensions put more burden on the nodes, since now they also need to verify the initiator's signature in each route request message and to maintain information that

is required by the rate limiting mechanism.

Finally, I note that, similarly to SRDP in [Kim and Tsudik, 2005], endairA can be optimized with respect to communication overhead by replacing the signature list in the route reply with a single aggregate or multi-signature (e.g., [Boneh *et al.*, 2003]) computed by the intermediate nodes iteratively in a similar way as in the case of the iterated MAC technique in the optimized version of Ariadne [Hu *et al.*, 2005] and in SRDP. In [Kim and Tsudik, 2005], the authors adapted several multi-signature schemes to authenticate reply messages in dynamic source routing, and they also compared that schemes in terms of computational overhead.

3.6 Summary

In this chapter, I showed that Ariadne is an insecure routing protocol regarding a minimal security objective of secure source routing protocols. Particularly, mounting a routing state pollution attack, the adversary can achieve that Ariadne returns non-plausible routes with non-negligible probability. Hence, I designed a novel secure dynamic source routing protocol called endairA which is provably secure secure considering the same security objective. Therefore, I successfully demonstrated the applicability of my model in two ways. First, I proved that a real routing protocol is secure in my model. Second, I showed that my model can be used to distinguish between routing protocols in terms of security.

Besides being provably secure, endairA has another significant advantage over Ariadne (and similar protocols): it is more efficient, because, similarly to SRDP, it requires less cryptographic computation overall from the nodes. This is because in endairA, only the processing of the route reply messages involves cryptographic operations, and a route reply message is processed only by those nodes that are in the node list carried in the route reply. In contrast to this, in Ariadne, the route request messages need to be digitally signed by all intermediate nodes; however, due to the way a route request is propagated, this means that each node in the network must sign each and every route request.

Chapter 4

Secure Dynamic Distance Vector Routing in Wireless Ad hoc Networks

This chapter further demonstrates the usefulness of my model described in Chapter 2. In particular, I adapt my model to dynamic distance vector routing in wireless ad hoc networks by defining a general security objective of distance vector routing protocols. Then, I present two similar routing state pollution attacks against the secure variant of AODV called SAODV [Zapata and Asokan, 2002]. Finally, I prove that ARAN [Sanzgiri et al., 2002], proposed independently from my work, is a secure distance vector routing protocol in my model. The fact that SAODV is insecure in the same model proves that my model is capable of distinguishing between secure distance vector routing protocols.

SAODV [Zapata and Asokan, 2002] is a "secure" variant of the Ad hoc On-demand Distance Vector (AODV) [Perkins and Royer, 1999] routing protocol. Authenticated Routing for Ad hoc Networks (ARAN) is another secure, distance vector routing protocol for ad hoc networks proposed in [Sanzgiri *et al.*, 2002]. In this chapter, I show that ARAN is secure with respect to a general security objective of distance vector routing protocols. This objective formalizes the requirement that every routing entry of all honest nodes must be "correct". Informally, a routing entry with a given destination node and cost value is correct, if it points to a neighboring node from where there exists a "workable" route towards the destination node with the given cost in the network. In contrast to this, SAODV is insecure regarding this objective as the attacks in Subsection 4.4 can cause incorrect entries to be stored at honest nodes.

The outline of this chapter is the following. Section 4.1 describes the operation of SAODV. Section 4.2 defines the security objective of distance vector routing in wireless ad hoc networks. In Section 4.3, I describe the tolerable imperfections of my model. In Section 4.4, I show that SAODV is insecure considering this security objective, whereas, in Section 4.5, I prove that ARAN is secure considering the same security objective. Finally, in Section 4.6, I conclude the chapter.

4.1 Operation of SAODV

The operation of SAODV is similar to that of AODV, but it uses cryptographic extensions to provide integrity of routing messages and to prevent the manipulation of the hop count information. Conceptually, SAODV routing messages (i.e., route requests and route replies) have a non-mutable and a mutable part. The non-mutable part includes, among other fields, the node sequence numbers, the addresses of the source and the destination, and a request identifier, while the mutable part contains the hop count information. Different mechanisms are used to protect the different parts.

The non-mutable part is protected by the digital signature of the originator of the message (i.e., the source or the destination of the route discovery). This ensures that the non-mutable fields cannot be changed by an adversary without the change being detected by the non-corrupted nodes.

In order to prevent the manipulation of the hop count information, the authors propose to use hash chains. When a node originates a routing message (i.e., a route reply or a route request), it first sets the HopCount field to 0, and the MaxHopCount field to the TimeToLive value. Then, it generates a random number *seed*, and puts it in the Hash field of the routing message. After that, it calculates the TopHash field by hashing *seed* iteratively MaxHopCount times. The MaxHopCount and the TopHash fields belong to the non-mutable part of the message, while the HopCount and the Hash fields are mutable. Every node receiving a routing message hashes the value of the Hash field (MaxHopCount − HopCount) times, and verifies whether the result matches the value of the TopHash field. Then, before rebroadcasting a route reply or forwarding a route request, the node increases the value of the HopCount field by one, and updates the Hash field by hashing its value once.

The rationale behind using the above hash chaining mechanism is that given the values of the Hash, the TopHash, and the MaxHopCount fields, anyone can verify the value of the HopCount field. On the other hand, preceding hash values cannot be computed starting from the value in the Hash field due to the one-way property of the hash function. This ensures that an adversary cannot decrease the hop count, and thus, cannot make a route appearing shorter than it really is. However, as we will see later (and as pointed out by the authors of SAODV themselves), this latter statement does not hold in general, because a corrupted node that happens to be on a route between the source and the destination may pass on the routing message without increasing the value of the HopCount field and without updating the value of the Hash field.

4.2 Security objective

I assume that an entry of the routing table of a given node v contains the following three fields: the identifier of the target node, the identifier of the next hop towards the target, and the cost value that represents the believed cost of the route to the given target via the given next hop. Without loss of generality, I assume that the routing metric is such that routes with lower cost values are preferred.

Consequently, the state of the system in my model will be represented by a set $Q \subset (V \setminus V^*) \times L \times L \times \mathbb{R}$ of quadruples such that for any (v, z_{tar}, z_{nxt}, c) and $(v', z'_{tar}, z'_{nxt}, c')$ in Q, $v = v'$ and $z_{tar} = z'_{tar}$ and $z_{nxt} = z'_{nxt}$ implies $c = c'$. The quadruple (v, z_{tar}, z_{nxt}, c) in Q represents an entry in v's routing table with target identifier z_{tar}, next hop identifier z_{nxt}, and believed route cost c. The ensemble of quadruples that have v as their first element represent the entire routing table of v, and the ensemble of all quadruples in Q represent the ensemble of the routing tables of all honest nodes (i.e., the state of the system). Note that we allow that a node's routing table contains multiple entries for the same target, but the next hops should be different.

Considering that SAODV uses the hop count, and ARAN uses the message propagation delay (i.e., physical time) as a path length metric, $\mathcal{C} : V \to \mathbb{R}$ assigns a constant 1 to each node in case of SAODV, while it assigns the minimal delay of routing messages to each node in the network (i.e., the minimal delay that the particular node can cause in the travel of the message) in case of ARAN. In order to ease further formalizations, I introduce the definition of workable path.

Definition 4 (Workable path) *A sequence of honest nodes $(v_{\ell_0}, v_{\ell_1}, \ldots, v_{\ell_{d-1}}, v_{\ell_d})$ is a workable path with respect to configuration conf if for all $0 \leq i \leq d-1$ v_{ℓ_i} and $v_{\ell_{i+1}}$ are direct or pseudo neighbors.*

I define correct states as follows:

Definition 5 (Correct state) *A state Q is correct if for every entry $(v_{src}, z_{dest}, z_{nxt}, c_{src}) \in Q$, there exists a sequence of entries $(v_{\ell_i}, z_{dest}, z_{\ell_i}, c_{\ell_i}) \in Q$ $(1 \leq i \leq d)$ such that*

- *$(v_{src}, v_{\ell_1}, \ldots, v_{\ell_{d-1}}, v_{dest})$ is a workable path, where $v_{\ell_d} = v_{dest}$,*
- *$z_{dest} \in \mathcal{L}(v_{dest})$,*
- *let $v_{\ell_0} = v_{src}$ and $z_{\ell_0} = z_{nxt}$,*
  - *if $v_{\ell_{i-1}}$ and v_{ℓ_i} are direct but not pseudo neighbors then $z_{\ell_{i-1}} \in \mathcal{L}(v_{\ell_i})$,*
  - *if $v_{\ell_{i-1}}$ and v_{ℓ_i} are pseudo neighbors then either $z_{\ell_{i-1}} \in \mathcal{L}(v_{\ell_i})$, or $z_{\ell_{i-1}} \in L^*$,*
- *$\sum_{i=1}^{d-1} \mathcal{C}(v_{\ell_i}) \leq c_{src}$.*

Intuitively, the system is in a correct state, if all the routing table entries of the *honest* nodes are correct in the sense that if v_{src} has an entry for target z_{dest} with next hop z_{nxt} and cost c_{src}, then indeed there exists a route in the network that

- starts from node v_{src},
- ends at a node that uses the identifier z_{dest},
- has a cost that is smaller than or equal to c_{src}, and
- all consecutive honest nodes lying on the route have a next-hop identifier towards v_{dest} which is either a corrupted identifier, or the identifier of the next direct or pseudo neighboring honest node.

Let the security objective function $\mathcal{F}$ of secure dynamic distance vector routing return 0 for all pairs of system states and configurations where the system state is incorrect with respect to the configuration. Otherwise, $\mathcal{F}$ returns 1.

4.3 Tolerable imperfections

Similarly to secure source routing in the previous chapter, I also use the notion of pseudo neighbors in Definition 5. The explanation is the same as it was in Section 3.3. In particular, two adversarial nodes that are located on different parts of the network can transfer the signatures of honest nodes by using out-of-band (e.g., wormholes) or in-band channels (e.g., hidden channel attacks).

The third point in Definition 5 requires that if the next node on the route is a direct but not pseudo neighbor then the next-hop identifier of the entry must be used by that next node, otherwise the next-hop identifier must be compromised. For instance, let us see two nodes $v_{\ell_{i-1}}, v_{\ell_i}$ on the discovered workable path. It is clear that if $v_{\ell_{i-1}}$ and v_{ℓ_i} are not direct neighbors, but they are pseudo neighbors then the adversary can modify the message received from v_{ℓ_i} at her own wish before sending that to $v_{\ell_{i-1}}$. However, the best that she can achieve is that v_{ℓ_i} sets a compromised next-hop identifier towards the destination. Now, let us assume that $v_{\ell_{i-1}}$ and v_{ℓ_i} are direct neighboring nodes on the discovered workable path. In that case, it is easy to see that if only one of them has an adversarial neighbor, then the adversary cannot modify the message coming from v_{ℓ_i}, as either she cannot hear v_{ℓ_i} or she cannot send the message to $v_{\ell_{i-1}}$. If $v_{\ell_{i-1}}$ and v_{ℓ_i} are direct neighbors and both of them have an adversarial neighbor, then they can hear each other, but the adversary can prevent v_{ℓ_i} from receiving the message coming from v_{ℓ_i} (e.g., by jamming), and then she can send the modified message to $v_{\ell_{i-1}}$. On the other hand, the adversary can only force v_{ℓ_i} to use a compromised next-hop identifier towards the destination. As we do not intend to defend against malicious packet dropping and jamming, we also tolerate these attacks.

In addition, the requirement on the believed cost of the route (last point in Definition 5) also introduces a new tolerable imperfection in the model. First of all, recall the assumption that routes with a lower cost are preferred. It is, therefore, natural to assume that the adversary wants to make routes appearing less costly than they are. This means that if node v_{src} believes that there exists a route between itself and target z_{dest} (passing through direct or pseudo neighbor z_{nxt}) with a cost c_{src}, while in reality, there exist only routes between them with a cost higher than c_{src}, then the system should certainly be considered to be in an incorrect state (i.e., under attack). On the other hand, allowing the existence of routes with a smaller cost does not have any harm (under the assumption that the adversary has no incentive to increase the believed costs corresponding to the routes), and it makes the definition of the correct state less demanding. This has a particular importance in case of protocols that use one-way hash chains to protect hop count values (e.g., SAODV and alike), since in those protocols, the adversary can always increase the hop count by hashing the current hash chain element further. However, this ability of the attacker should rather be viewed as a tolerable imperfection of the system than a flaw in those protocols.

4.4 Insecurity of SAODV

In the following, I show that SAODV is not secure in my model. Particularly, I show that SAODV cannot guarantee that the next hop and the hop count information in the newly created routing table entry is correct.

Theorem 4 *SADOV is an insecure distance vector routing protocol for wireless ad hoc networks.*

Proof 1 I show that there exists a configuration $conf'$ and an adversary $\mathcal{A}$ such that an incorrect state occurs with non-negligible probability.

Let us consider configuration $conf'$ illustrated in Figure 4.1.

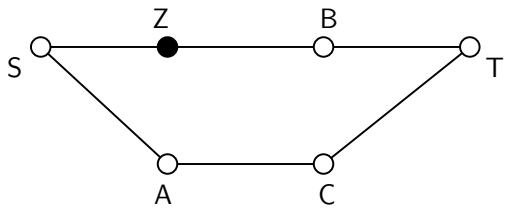


Figure 4.1: A network topology where the adversary can achieve that the node labelled by T creates in its routing table an entry with an incorrect hop-count value when SAODV is used. Honest nodes are labelled by S, A, C, T, and B, whereas the single adversarial node is labelled by Z.

Non-adversarial machines are referred to by their identifiers. Let us assume that S starts a route discovery towards T. When the route request message reaches the adversarial machine (which represents node Z), it does not increase the hop count and does not update the hash value in the message. Therefore, when this route request is eventually received by T, it will believe that there is a route towards S passing through neighbor B and has a length 1. In addition, the corresponding entry will not be overwritten when the other route request message arrives through C, since that request will have a hop count of 2. However, there is not any route in this network that starts at the node labelled by T, passes through the node labelled by B, ends at the node labelled by S, and has a length less than or equal to 1. ∎

I note that this weakness of SAODV has already been known by its authors (see Subsection 5.3.5 of [Zapata and Asokan, 2002]).

Proof 2 I show that there exists another configuration $conf''$ and another adversary $\mathcal{A}'$, which are different from $conf'$ and $\mathcal{A}$, resp., such that an incorrect state occurs with non-negligible probability.

Let us now consider the configuration illustrated in Figure 4.2. Let us assume again that the source is S and the destination is T. Furthermore, let us assume that a route request message reached the destination, and it returned an appropriate route reply. When this reply reaches the adversarial machine, it forwards it to S in the name of A. Therefore, S will believe

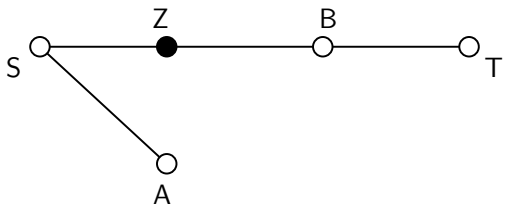


Figure 4.2: A network topology where the adversary can achieve that the node labelled by S creates in its routing table an entry for target T with an incorrect next hop A when SAODV is used. Honest nodes are labelled by S, A, T, and B, whereas the single adversarial node is labelled by Z.

that there is a route towards T passing next-hop A and has a length of 2 hops. Note, however, that there is no route at all from the node labelled by S to the node labelled by T that passes through the node labelled by A. ■

To the best of my knowledge, the last weakness of SAODV has been first published in [Ács *et al.*, 2005a].

4.5 Security of ARAN

Compared to SAODV, ARAN uses previous-hop (i.e., neighboring) authentication by requiring each intermediate hop to sign the request and reply message. This signature is then updated by the next hop. Thus, the message size remains constant, and each intermediate hop can check whether the received message is indeed sent by a neighboring node. Another difference between SAODV and ARAN is the routing metric used to select among discovered routes. While SAODV applies hop-count to rank discovered routes, ARAN uses time (i.e., network delay) for this purpose.

In this section, I analyse the security of ARAN. In Subsection 4.5.1, I give a brief overview of the operation of ARAN. Subsection 4.5.2 contains the security proof of ARAN.

4.5.1 Operation of ARAN

Just like SAODV, ARAN [Sanzgiri *et al.*, 2002] as well uses public key cryptography to ensure the integrity of routing messages. Initially, a source node S begins a route discovery process by broadcasting a route request message:

$$(\mathsf{rreq}, T, cert_S, N_S, t, sig_S)$$

where rreq means that this is a route request, S and T are the identifiers of the source and the destination, respectively, N_S is a nonce generated by S, t is the current time-stamp, $cert_S$ is the public-key certificate of the source, and sig_S is the signature of the source on all of these elements. N_S is a monotonically increasing value that, together with t and S, uniquely

identifies the message, and it is used to detect and discard duplicates of the same request (and reply).

Later, as the request is propagated in the network, intermediate nodes also sign it. Hence, the request has the following form in general:

$$(\mathsf{rreq}, T, cert_S, N_S, t, sig_S, sig_A, cert_A)$$

where A is the identifier of the intermediate node that has just re-broadcast the request. When a neighbor of A, say B, receives this route request, then it verifies both signatures, and the freshness of the nonce. If the verification is successful, then B sets an entry in its routing table with S as target, and A as next hop. Then, B removes the certificate and the signature of A, signs the request, appends its own certificate to it, and rebroadcasts the following message:

$$(\mathsf{rreq}, T, cert_S, N_S, t, sig_S, sig_B, cert_B)$$

When destination T receives the first route request that belongs to this route discovery, it performs verifications and updates it routing table in a similar manner as it is done by the intermediate nodes. Then, it sends a route reply message to S. The route reply is propagated back on the reverse of the discovered route as a unicast message. The route reply sent by T has the following form:

$$(\mathsf{rrep}, S, cert_T, N_S, t, sig_T)$$

where rrep means that this is a route reply, N_S and t are the nonce and the time-stamp obtained from the request, S is the identifier of the source, $cert_T$ is the public-key certificate of T, and sig_T is the signature of T on all of these elements.

Similar to the route request, the route reply is signed by intermediate nodes too. Hence, the general form of the route reply is the following:

$$(\mathsf{rrep}, S, cert_T, N_S, t, sig_T, sig_B, cert_B)$$

where B is the identifier of the node that has just passed the reply on.

A node A that receives the route reply verifies both signatures in it, and if they are valid, then it forwards the reply to the neighbor node from which it has received the corresponding route request previously. However, before doing that, A will remove the certificate and the signature of B, and put its own certificate and signature in the message:

$$(\mathsf{rrep}, S, cert_T, N_S, t, sig_T, sig_A, cert_A)$$

In addition, A also sets an entry in its routing table for target T with B as the next hop.

As it can be seen from the description, ARAN does not use hop counts as a routing metric. Instead, the nodes update their routing tables using the information obtained from the routing messages that arrive first; any later message that belongs to the same route discovery is discarded. This means that ARAN may not necessarily discover the shortest paths in the network, but rather, it discovers the quickest ones. In effect, ARAN uses the

message propagation delay (i.e., physical time) as a path length metric.

4.5.2 Security proof

Theorem 5 *ARAN is a secure distance vector routing protocol for wireless ad hoc networks, if the signature scheme is secure against chosen message attacks.*

Proof (sketch) Since ARAN uses the message propagation delay as the routing metric, I will assume that the node cost values in my model represent minimum message processing delays (at the nodes). In addition, I make the pessimistic assumption that the adversary's message processing delay is 0, which means that $\mathcal{C}(v) = 0$ for all $v \in V^*$.

In order to be compliant with my framework, I also assume that each routing table entry explicitly contains a routing metric value too. In our case, this metric value is the time that was needed for the routing message that triggered the creation of this entry to get from the originator of the message to the node that created this entry. Although these times are not represented explicitly in ARAN routing table entries, representing them in the model does not weaken my results in any way. In particular, exactly the same routing table entries are created in my model as in ARAN with respect to the target and the next hop identifiers.

In order to prove that ARAN is secure, according to Definition 1, one has to show that the system encounters incorrect states only with a probability that is negligible function of κ, where κ is practically the key length of the applied signature scheme. Particularly, if no incorrect state is encountered during the computation of the model, then $\mathcal{F}$ always returns 1 which means that the protocol is secure. On the other hand, if an incorrect state occurs in the model, the protocol can still be secure according to Definition 1, if this happens only with negligible probability. I will show that indeed this is the case for ARAN.

Getting into an incorrect state means that one of the honest nodes sets an incorrect entry in its routing table. In particular, the system running ARAN encounters incorrect state in the cases as follows:

- Case 1: There exists an entry $(v_{src}, z_{dest}, z_{nxt}, c_{src})$, but there does not exist a workable path between v_{src} and a node which uses z_{dest}.
- Case 2: There exists an entry $(v_{src}, z_{dest}, z_{nxt}, c_{src})$ and at least one workable path between v_{src} and z_{dest} where $z_{dest} \in \mathcal{L}(v_{dest})$, but there also exists a pair of consecutive honest nodes $v_{\ell_{i-1}}, v_{\ell_i}$ on each workable path such that
  - either $v_{\ell_{i-1}}, v_{\ell_i}$ are direct but not pseudo neighbors and $z_{\ell_{i-1}} \notin \mathcal{L}(v_{\ell_i})$,
  - or $v_{\ell_{i-1}}, v_{\ell_i}$ are pseudo neighbors but neither $z_{\ell_{i-1}} \in \mathcal{L}(v_{\ell_i})$, nor $z_{\ell_{i-1}} \in L^*$,

  where $z_{\ell_{i-1}}$ is the next-hop identifier in entry $(v_{\ell_{i-1}}, z_{dest}, z_{\ell_{i-1}}, c_{\ell_{i-1}})$.
- Case 3: There exists an entry $(v_{src}, z_{dest}, z_{nxt}, c_{src})$ and at least one workable path between v_{src} and v_{dest} where all subsequent honest nodes lying on each path have a next-hop identifier towards v_{dest} which is either a corrupted identifier, or the identifier of the next direct or pseudo neighboring honest node, however, c_{src} on each route is less than the real cost of the route.

In case 1, if the signature of z_{dest} in the routing message is not forged, then the very fact that v_{src} received the message proves that there is a workable path between v_{src} and a node that uses z_{dest} (since otherwise the message could not reach v_{src}). Hence, case 1 is possible only if the signature of z_{dest} is forged, and this has negligible probability if the signature scheme is secure.

In case 2, let us assume that the adversary cannot forge any signatures. As v_{src} has entry $(v_{src}, z_{dest}, z_{nxt}, c_{src})$, v_{src} receives either a RREP or a RREQ message with a correct signature of v_{dest}. Thus, based on Case 1, there exists a workable path between v_{src} and v_{dest} along which the request (or reply) message, denoted by msg, is received by v_{src}. According to our assumption, two further cases can be distinguished:

- Case 2a: there exists i such that $v_{\ell_{i-1}}, v_{\ell_i}$ are direct but not pseudo neighbors, however, $z_{\ell_{i-1}} \notin \mathcal{L}(v_{\ell_i})$.
- Case 2b: there exists i such that $v_{\ell_{i-1}}, v_{\ell_i}$ are pseudo neighbors, however, $z_{\ell_{i-1}} \notin \mathcal{L}(v_{\ell_i})$ as well as $z_{\ell_{i-1}} \notin L^*$.

In case 2a, as $sig_{z_{\ell_i}}$ can only be generated by v_{ℓ_i}, $v_{\ell_{i-1}}$ received an msg' message ($msg' \neq msg$) with previous-hop signature sig_{z_x}, where $sig_{z_x} \neq sig_{z_{\ell_i}}$. Since $sig_{z_{dest}}$ travelled through a workable path between v_{src} and v_{dest}, z_x belongs to an adversarial node and the adversary obtained $sig_{z_{dest}}$ from v_{ℓ_i}. Therefore, both $v_{\ell_{i-1}}$ and v_{ℓ_i} have an adversarial neighbor, which means that they are pseudo neighbors. However, this contradicts to our assumption that $v_{\ell_{i-1}}$ and v_{ℓ_i} cannot be pseudo neighbors. In case 2b, first let us assume that $z_{\ell_{i-1}} \notin L^*$. Thus, sig_{z_x} can only be generated by an honest node v'. As $sig_{z_{dest}}$ travelled through a workable path between v_{src} and v_{dest}, $v' = v_{\ell_i}$ which means that $z_{\ell_{i-1}} \in \mathcal{L}(v_{\ell_i})$. Similarly, based on case 2a, assuming $z_{\ell_{i-1}} \notin \mathcal{L}(v_{\ell_i})$ implies that $z_{\ell_{i-1}} \in L^*$. Consequently, case 2 occurs only if the adversary successfully forges a signature.

Finally, in case 3, let R be the set of existing workable paths that start at v_{src}, end at a node that uses z_{dest}, and all subsequent honest nodes lying on these paths have a next-hop identifier towards v_{dest} which is either a corrupted identifier, or the identifier of the next honest node. Moreover, let c' be the minimum of the costs of the routes in R. By assumption, $c' > c_{src}$. If the signatures of z_{dest} and z_{nxt} in the routing message received by v_{src} are not forged, then the message must have taken one of the routes in R. However, it could not reach v_{src} in time $c_{src} < c'$, since the node costs represent the minimum message processing and transmission delays at the nodes. In other words, the adversary cannot speed up the transmissions on the links and the processing at the non-corrupted nodes. Hence, case 3 is possible only if either z_{dest} or z_{nxt}, or both are forged, which can happen only with negligible probability. ∎

4.6 Summary

This chapter addressed the security of distance vector routing protocols in wireless ad hoc networks. First, after presenting the operation of SAODV, I adapted my model described in

Chapter 2 to distance vector routing by defining a general security objective of distance vector routing. Then, I proved that SAODV is insecure with respect to this security objective. Indeed, using SAODV, an adversarial node can cause incorrect entries to be set in honest nodes' routing table by mounting the attacks presented in Subsection 4.4. Then, after reviewing the operation of ARAN, I proved that it is, in turn, a secure distance vector routing protocol considering the same security objective. Therefore, my model can distinguish distance vector routing protocols in terms of security.

A conclusion of the analysis was that source and destination authentication in the route discovery process are not sufficient to guarantee security. In particular, without neighbor authentication a distance vector routing remains vulnerable to routing state pollution attacks.

Chapter 5

Secure Centralized Link-state Routing in Wireless Sensor Networks

This chapter demonstrates how my framework can be used to prove the security of wireless sensor network routing. Specifically, I adapt my model to secure link state routing in sensor networks by defining a general security objective of such routing protocols. Then, I formally prove that Intrusion-Tolerant Routing in Wireless Sensor Networks (INSENS), which is a secure sensor network routing protocol proposed in the literature independently of my work, can be proven to be secure in this adapted model.

INSENS [Deng *et al.*, 2002] is an intrusion-tolerant link-state routing protocol proposed for wireless sensor networks. The protocol consists of three operational phases. In Phase 1, each node determines its neighborlist by overhearing the route request flooded by the base station. When a node has learnt its neighborhood it sends its own neighborlist to the base station which is responsible for computing the routing table for all sensor nodes in the network in Phase 2. Finally, in Phase 3, the base station distributes the computed forwarding tables of all nodes in a secure way.

INSENS assumes that every node has a single symmetric key that is shared with the base station. As the base station uses a one-way hash chain to prevent malicious flooding, every node is additionally assumed to store the last element of the chain created by the base station. Moreover, bidirectional communication channels are assumed between each pair of nodes. In this chapter, I further demonstrate the strength of my model by showing that INSENS is a provably secure link-state routing protocol for wireless sensor networks in my model.

The organisation of this chapter is as follows. In Section 5.1, I give a brief overview of the operation of INSENS. Section 5.2 defines the security objective of link-state routing in wireless sensor networks. Then, in Section 5.3, I detail the tolerable imperfections of this model. Section 5.4 describes the security proof of INSENS in this model. Finally, I conclude the chapter in Section 5.5.

5.1 Operation of INSENS

In the following, I describe each operational phase of INSENS. More detailed description can be found in [Deng *et al.*, 2002]. Note that I am only concerned with the topology (route) discovery mechanism of INSENS and not with the data forwarding mechanism. In order to preserve the readability of the thesis, node identifiers are denoted in the same way as the nodes. If it is required to emphasize the distinction between the nodes and their identifiers in a given context I use the labelling function for this purpose.

Calculation of neighborlist: The base station initiates the routing topology construction by flooding the network with a route request message, which has the following format:

$$v_0 \rightarrow * \ : \ (\mathsf{REQ}, \mathsf{hash}, [v_0])$$

where REQ is a constant message type identifier, hash is the next element of the hash chain in reverse direction, and v_0 identifies the base station. The hash chain mechanism is intended to provide authenticity and some defense against Denial-of-Service (DoS) attacks. Each node constructs its own neighborlist by overhearing the request messages sent by its neighbors.

Every subsequent node v_{ℓ_i} receiving request

$$(\mathsf{REQ}, \mathsf{hash}, [v_0, v_{\ell_1}, \ldots, v_{\ell_{i-1}}], \mathsf{MAC}^{\mathsf{REQ}}_{v_{\ell_{i-1}}})$$

verifies the correctness of hash and checks whether it is the first request containing hash. If it is the first one, then v_{ℓ_i} re-broadcasts the modified request, and stores $\mathsf{MAC}^{\mathsf{REQ}}_{v_{\ell_{i-1}}}$ in conjunction with $v_{\ell_{i-1}}$ locally. Before re-broadcasting, v_{ℓ_i} replaces $\mathsf{MAC}^{\mathsf{REQ}}_{v_{\ell_{i-1}}}$ in the request with $\mathsf{MAC}^{\mathsf{REQ}}_{v_{\ell_i}}$, which is the MAC generated by v_{ℓ_i} on list $[v_0, \ldots, v_{\ell_{i-1}}, v_{\ell_i}]$, REQ, and hash using the symmetric key shared with v_0. Finally, v_{ℓ_i} re-broadcasts the following request:

$$v_{\ell_i} \rightarrow * \ : \ (\mathsf{REQ}, \mathsf{hash}, [v_0, \ldots, v_{\ell_{i-1}}, v_{\ell_i}], \mathsf{MAC}^{\mathsf{REQ}}_{v_{\ell_i}})$$

Forwarding neighborlist towards the base station: If a node v_{ℓ_x} does not receive further request messages for a specified time, v_{ℓ_x} sends the following message to $v_{\ell_{x-1}}$ from which it received the first valid request:

$$v_{\ell_x} \rightarrow v_{\ell_{x-1}} \ : \ (\mathsf{NLIST}, \mathsf{hash}, \mathsf{MAC}^{\mathsf{REQ}}_{v_{\ell_{x-1}}}, v_{\ell_x}, \mathsf{Enc}_{v_{\ell_x}}(path_{v_{\ell_x}}, neighborlist_{v_{\ell_x}}), \mathsf{MAC}^{\mathsf{NLIST}}_{v_{\ell_x}})$$

where the elements of the message are as follows: NLIST is a constant message type identifier; hash is the hash value of the corresponding request message; $\mathsf{MAC}^{\mathsf{REQ}}_{v_{\ell_{x-1}}}$ is the MAC, called parent MAC[1], of $v_{\ell_{x-1}}$ sent in the corresponding request; v_{ℓ_x} is the identifier of the message originator; $\mathsf{Enc}_{v_{\ell_x}}(path_{v_{\ell_x}}, neighborlist_{v_{\ell_x}})$ is the neighborhood information and the path information of v_{ℓ_x} encrypted by the symmetric key shared with the base station; $neighborlist_{v_{\ell_x}}$

[1] In this context, parent node is the next-hop that forwards neighborhood information, and not measured data, towards the base station.

contains the identifiers of each neighboring node *and* their corresponding MACs received in Phase 1; $path_{v_{\ell_x}}$ is $[v_{\ell_x}, \ldots, v_{\ell_1}, v_0, \mathsf{MAC}^{\mathsf{REQ}}_{v_{\ell_x}}]$, which is the reverse of the path received in the corresponding request message including the MAC of node v_x; and finally $\mathsf{MAC}^{\mathsf{NLIST}}_{v_{\ell_x}}$ is the MAC computed by node v_{ℓ_x} on NLIST, hash, $path_{v_{\ell_x}}$, and $neighborlist_{v_{\ell_x}}$.

A node receiving the reply message first checks if the node is the parent of the sender (i.e., $\mathsf{MAC}^{\mathsf{REQ}}_{v_{\ell_{x-1}}}$ message equals to its own MAC that has been broadcast with request containing hash). Then, the node replaces the parent MAC in the message with its own parent MAC that is stored in Phase 1. In this way, the reply message propagates back to the base station. Upon the reception of a reply message

$$(\mathsf{NLIST}, \mathsf{hash}, v_{\ell_x}, \mathsf{Enc}_{v_{\ell_x}}(path_{v_{\ell_x}}, neighborlist_{v_{\ell_x}}), \mathsf{MAC}^{\mathsf{NLIST}}_{v_{\ell_x}})$$

the base station checks whether all the MACs are correct, after decrypting $\mathsf{Enc}_{v_{\ell_x}}(path_{v_{\ell_x}}, neighborlist_{v_{\ell_x}})$[2]. If all verifications are successful, the base station computes the forwarding table for each node using a global centralized algorithm detailed in [Deng *et al.*, 2002].

Distributing forwarding tables: The forwarding tables are propagated to respective nodes in a breadth-first manner; first, the immediate neighbors of the base station receive their forwarding tables directly from the base station. Afterwards, these one-hop neighbors forward the forwarding tables of the two-hop neighbors of the base station based on their forwarding tables, and so on. In particular, the base station first sends the forwarding table of v_{ℓ_1}:

$$v_0 \rightarrow v_{\ell_1} \; : \; (\mathsf{FTABLE}, v_{\ell_1}, \mathsf{hash}, \mathsf{Enc}_{v_{\ell_1}}(ftable_{v_{\ell_1}}), \mathsf{MAC}^{\mathsf{FTABLE}}_{v_{\ell_1}})$$

where FTABLE is a constant message type identifier, $\mathsf{Enc}_{v_{\ell_1}}(ftable_{v_{\ell_1}})$ is the encrypted form of the forwarding table of v_{ℓ_1}, and $\mathsf{MAC}^{\mathsf{FTABLE}}_{v_{\ell_1}}$ is the MAC generated by v_0 on the complete message. Upon the reception of this message, v_{ℓ_1} sets its forwarding rules according to $ftable_{v_{\ell_1}}$, if $\mathsf{MAC}^{\mathsf{FTABLE}}_{v_{\ell_1}}$ is correct.

5.2 Security objective

Similarly to authenticated TinyOS beaconing in Subsection 2.2.7, the system state with a given configuration *conf* is represented by a matrix T^{conf} (or simply T) with size $(k+1)\times(k+2)$, and which is called as a *routing topology* with configuration *conf*. I recall that, for all $0 \leq i \leq k$,

- and $0 \leq j \leq k$, $T_{i,j} = 1$, if honest node v_i sends every message to an honest node v_j in order to deliver the message to the destination node, otherwise let $T_{i,j}$ be 0.

- $T_{i,k+1} = 1$, if honest node v_i sends every message to an adversarial node in order to deliver the message to the destination node, otherwise let $T_{i,j}$ be 0.

[2] Actually, the MACs in the $neighborlist_{v_{\ell_x}}$ can only be checked when the NLIST messages of the corresponding nodes in $neighborlist_{v_{\ell_x}}$ are also received.

I show that INSENS described in Section 5.1 is a secure link-state routing protocol in my model. I show that the protocol has the following properties:

1. If an honest sensor node v_i $(1 \leq i \leq k)$ sets $v_j \in V$ $(0 \leq j \leq n-1)$ as its parent node for data forwarding, then the base station has indeed computed v_j as the parent node for v_i.

2. If the base station is aware of the fact that node v_j is a neighbor of node v_i, then node v_i and v_j are direct or pseudo neighbors.

Intuitively, if INSENS has these two properties, then it is ensured that each honest node has a *neighboring* parent node that is computed by the base station. Moreover, it is also guaranteed that this computation performed by the base station is based on, perhaps incomplete (the adversary can always drop routing messages containing neighborlists, which we are unable to defend against), but correct neighborhood information. In fact, this is a general security objective of every kind of centralized link-state routing protocol for sensor networks.

In order to formalize the above security objective, we introduce a matrix function $\mathcal{G}$. $\mathcal{G}$ models the centralized construction of the topology performed by the base station, where the argument of $\mathcal{G}$ with size $(k+1) \times (k+2)$, denoted by $\mathbf{N}$, describes the neighborhood relations among the sensor nodes which is believed by the base station to be correct (i.e., $N_{i,j} = 1$ if the base station believes that v_i is a neighbor of v_j, otherwise $N_{i,j} = 0$. For any $0 \leq i \leq k$, $N_{i,k+1} = 1$, if honest node v_i has at least one adversarial neighbor, otherwise $N_{i,k+1} = 0$). The output of $\mathcal{G}$ is the ensemble of the routing entries (the routing topology) that should be set by each node.

Definition 6 (Correct routing topology) *A routing topology* $\mathbf{T}$ *is correct with respect to configuration* $conf$*, if there exists a matrix* $\mathbf{E}'$ *such that for all* i, j *it holds that if* $T_{i,j} = 1$*, then* $\mathcal{G}(\mathbf{E}')_{i,j} = 1$*, where* $\mathbf{E}'$ *is derived from* $\mathbf{E}$ *with size* $(k+1) \times (k+2)$ *and it is defined as follows. For all* $0 \leq i, j \leq k$*,* $E'_{i,j} = 0$*, if* v_i *and* v_j *are neither direct nor pseudo neighbors. For all* $0 \leq i \leq k$*,* $E'_{i,k+1} = 0$*, if* v_i *has no direct adversarial neighbor.*

Let the security objective function $\mathcal{F}$ of secure link-state routing return 0 for all pairs of system states and configurations where the system state (i.e., the routing topology) is incorrect with respect to the configuration. Otherwise, $\mathcal{F}$ returns 1.

5.3 Tolerable imperfections

Observe that the reduction of a reachability matrix $\mathbf{E}$ to $\mathbf{E}'$ is not unambiguous in Definition 6. Particularly, we only require those pairs of nodes to be non-neighbors in $\mathbf{E}'$ that are also non-neighbors in $\mathbf{E}$. The rationale is that, based on the adversary model in Section 2.1.4, the adversary may be able to break links between honest nodes by deleting some or all messages on that. As message deletion is unavoidable or at least it is too costly to defend against, I also consider this as a tolerable imperfection of my model.

Furthermore, similarly to Sections 3.2 and 4.2, the adversary can also install in-band as well as out-of-band channels. For instance, the encrypted neighborlist information in messages

NLIST can be exploited to transfer the authentication information of neighboring honest nodes to remote adversarial nodes which is a type of in-band channel attacks. This attack is similar to the hidden channel attack detailed in Section 3.3, as the adversary also sends the authentication information of its honest neighborhood (i.e., their $\mathsf{MAC}^{\mathsf{REQ}}$s) to remote adversarial parties as a part of an authentic routing message (recall that the adversary is insider). Therefore, I also use the notion of pseudo neighbors in Definition 6 to exclude these types of attacks from my model.

5.4 Security proof

Theorem 6 *INSENS is a secure link-state routing protocol for wireless sensor networks, if the MAC scheme is secure against chosen message attack, and the symmetric encryption scheme is secure against plaintext recovery attack.*

Proof (sketch) I show that for any adversary $\mathcal{A}$ and any configuration $conf$, $\mathcal{F}(conf, \mathbf{T}) = 0$ only with probability that is a negligible function of κ_1 and κ_2, where κ_1, κ_2 are the security parameters of the employed MAC and encryption schemes, resp. In other words, the success probability of any adversary is a negligible function of κ_1 and κ_2.

From the definition of $\mathcal{F}$, $\mathcal{F}(conf, \mathbf{T}) = 0$ if there exist i, j $(1 \leq i \leq k, 0 \leq j \leq k+1)$ such that $T_{i,j} = 1$ and there does not exist any $\mathbf{E}'$, such that $\mathcal{G}(\mathbf{E}')_{i,j} = 1$. This can have two reasons as follows: (i) node v_i received incorrect routing topology information, or (ii) the base station received incorrect neighborhood information. According to this, I introduce the following events:

(i) $\mathsf{C}_1^{i,j}$ denotes the event that $T_{i,j} = 1$, but $\mathcal{G}(\mathbf{N})_{i,j} = 0$,

(ii) $\mathsf{C}_2^{i,j}$ denotes the event that $T_{i,j} = 1$, $\mathcal{G}(\mathbf{N})_{i,j} = 1$, and $N_{i,j} = 1$, but v_i and v_j are neither direct nor pseudo neighbors.

I recall that $\mathbf{N}$ describes the neighborhood relations among the sensor nodes, which is believed by the base station to be correct. Clearly, the following upper estimation holds for the success probability of the adversary denoted by $P^{\mathcal{A}}$:

$$P^{\mathcal{A}} \leq \sum_{\forall i,j: i \neq j, i \neq 0} \mathbf{P}\left(\mathsf{C}_1^{i,j}\right) + \sum_{\forall i,j: i \neq j, i \neq 0} \mathbf{P}\left(\mathsf{C}_2^{i,j}\right)$$

I show that $\mathbf{P}\left(\mathsf{C}_1^{i,j}\right)$ is a negligible function of κ_1, and $\mathbf{P}\left(\mathsf{C}_2^{i,j}\right)$ is a negligible function of κ_1 and κ_2 for all i, j. This implies that $P^{\mathcal{A}}$ is also a negligible function of κ_1 and κ_2 that concludes the theorem.

Negligibility of $\mathbf{P}\left(\mathsf{C}_1^{i,j}\right)$: If $\mathsf{C}_1^{i,j}$ occurs, then M_i receives an FTABLE message, which contains the routing information of node v_i:

$$(\mathsf{FTABLE}, v_i, \mathsf{hash}, \mathsf{Enc}_{v_i}(ftable'_{v_i}), \mathsf{MAC}'^{\mathsf{FTABLE}}_{v_i})$$

v_i infers from $ftable'_{v_i}$ that $T_{i,j} = 1$, since $\mathsf{MAC}'^{\mathsf{FTABLE}}_{v_i}$ is a correct MAC. I show that it is only possible if $\mathsf{MAC}'^{\mathsf{FTABLE}}_{v_i}$ is a successfully forged MAC by A.

Let us assume that A cannot forge $\mathsf{MAC}'^{\mathsf{FTABLE}}_{v_i}$. Hence, M_0 is the only machine who generates $\mathsf{MAC}'^{\mathsf{FTABLE}}_{v_i}$. However, M_0 generates $\mathsf{MAC}'^{\mathsf{FTABLE}}_{v_i}$ only if $[\mathcal{G}(N)]_{i,j} = 1$, which is a contradiction.

Consequently, $\mathsf{C}_1^{i,j}$ occurs for any i, j, if the adversary $\mathcal{A}$ successfully forges a MAC. However, the probability of this event is a negligible function of κ_1 assuming that $\mathcal{A}$ runs in polynomial time.

Negligibility of P $\left(\mathsf{C}_2^{i,j}\right)$: If $\mathsf{C}_2^{i,j}$ occurs, then M_0 receives an NLIST message, which contains the neighborhood information of node v_j:

$$(\mathsf{NLIST}, \mathsf{hash}, v_j, \mathsf{Enc}_{v_j}(path_{v_j}, neighborlist'_{v_j}), \mathsf{MAC}'^{\mathsf{NLIST}}_{v_j})$$

v_0 infers from $neighborlist'_{v_j}$ that $N_{i,j} = 1$, since $\mathsf{MAC}'^{\mathsf{NLIST}}_{v_j}$ is a correct MAC. I show that it is only possible if at least one of the following conditions holds:

1. $\mathsf{MAC}'^{\mathsf{NLIST}}_{v_j}$ is a successfully forged MAC by A, if v_j is an honest node.
2. There exists a node v_t $(1 \leq t \leq k)$ which is a direct neighbor of v_i, and A successfully recovered the plaintext from $\mathsf{Enc}_{v_t}(path_{v_t}, neighborlist_{v_t})$ that is sent in the corresponding NLIST message by v_t.
3. $\mathsf{MAC}'^{\mathsf{REQ}}_{v_i}$ that is received by v_j is a successfully forged MAC by A.

Let us assume that *none* of the above conditions hold. Two main cases can be distinguished: (i) v_j is an honest node, or (ii) v_j is an adversarial node.

(i) Based on the argument of the negligibility of $\mathsf{C}_1^{i,j}$, we know that $\mathsf{MAC}'^{\mathsf{NLIST}}_{v_j}$ can only be generated by M_j. Thus, M_j received a REQ message denoted by

$$msg' = (\mathsf{REQ}, \mathsf{hash}, [v_0, \ldots, v_i], \mathsf{MAC}'^{\mathsf{REQ}}_{v_i})$$

We know that msg' is never relayed by machines $M_0, \ldots, M_{i-1}, M_{i+1}, \ldots, M_k$, since these machines never send any REQ messages containing a path where the last element is v_i (such as path $[v_0, \ldots, v_i]$ in msg'). Therefore, M_j receives msg' from A implying that v_j has a direct adversarial neighbor.

Since v_i is not an adversarial node, $\mathsf{MAC}'^{\mathsf{REQ}}_{v_i}$ cannot be generated by machines $M_0, \ldots, M_{i-1}, M_{i+1}, \ldots, M_k, A$. Therefore, only M_i can generate $\mathsf{MAC}'^{\mathsf{REQ}}_{v_i}$. We know that msg' cannot be sent to M_j by M_i, since v_i and v_j are not direct neighbors. As v_j has an adversarial neighbor, v_i does not have any adversarial neighbor by assumption. Thus, in order to construct msg', A can only infer $\mathsf{MAC}'^{\mathsf{REQ}}_{v_i}$ from the messages sent by the neighbors v_t of v_i, since only honest nodes v_t can be reached by v_i, and these nodes only relay $\mathsf{MAC}'^{\mathsf{REQ}}_{v_i}$ in an encrypted form. In that case, $\mathsf{MAC}'^{\mathsf{REQ}}_{v_i}$ must be inferred from $\mathsf{Enc}_{v_t}(path_{v_{\ell_t}}, neighborlist_{v_t})$, which contradicts to our assumption.

(ii) Let us assume that v_i has no direct adversarial neighbor. Similarly to case (i), A can only infer $\mathsf{MAC}'^{\mathsf{REQ}}_{v_i}$ from the messages sent by the neighbors of v_i, as A is unable to forge $\mathsf{MAC}'^{\mathsf{REQ}}_{v_i}$. Thus, A must recover $\mathsf{MAC}'^{\mathsf{REQ}}_{v_i}$ from encrypted neighborlists. However, by assumption, the adversary cannot do this. This means that v_i has a direct adversarial neighbor, which is a contradiction again.

Consequently, $\mathsf{C}_2^{i,j}$ can only occur for any i, j, if at least one of the above conditions is true. This implies that the adversary $\mathcal{A}$ is able to forge a MAC, or $\mathcal{A}$ can recover the plaintext from a ciphertext. However, the probability of this event is a negligible function of κ_1 and κ_2 assuming that $\mathcal{A}$ runs in polynomial time. ∎

5.5 Summary

The link-state routing approach has some advantages in terms of security. First, intermediate nodes forwarding the link-state information do not need to access the packet in order to derive some routing state or update the routing information (e.g., hop counts) carried by the packet. This implicitly eliminates many potential attacks (e.g., hop-count manipulations). Second, many centralized countermeasures [Ács and Buttyán, 2008b] (like centralized wormhole detection schemes, registration techniques against replication attacks, anomaly detection against Sybil attacks, etc.) can be coupled with the central construction of a routing topology.

On the other hand, the applicability of INSENS is strongly constrained by its centralized nature. As the base station is the single point of failure in the network, it is exposed to DoS attacks even for an outsider adversary. Moreover, controlling strategic nodes (e.g., nodes that are close to the base station) the adversary can easily cause incomplete (but correct) routing topologies by simply dropping some NLIST, REQ and FTABLE messages.

Recall that the proof is strongly based on the assumption that the encryption scheme is secure against plaintext recovery attack. The encryption of neighborlists used in INSENS is crucial; apart from providing confidentiality for the neighborhood relations, the encryption of neighborlists prevents the adversary to impersonate honest nodes that are not covered by the transmission range of any adversarial nodes. For instance, if the neighborlists were not encrypted, an intermediate adversarial node could easily retrieve the identities and corresponding $\mathsf{MAC}^{\mathsf{REQ}}$s from NLIST messages, and then she could re-broadcast fabricated REQ messages. Note that the adversary is not required to reach the impersonated node directly. Apparently, this would also violate our security objective detailed in Subsection 5.2, as the adversary could cause the base station to consider false neighborhood relations. Furthermore, as $\mathsf{MAC}^{\mathsf{REQ}}$s are correct, it can happen that neither the neighbors of the adversary nor the base station could detect the misdeed. This attack scenario was not described in [Deng *et al.*, 2002], where the authors used informal reasoning to prove the security of INSENS. In contrast to this, my formal security analysis would reveal such flaw in a routing protocol: if encryption had not been employed, I could not have claimed in the proof that the adversary can retrieve the $\mathsf{MAC}^{\mathsf{REQ}}$ of a non-neighboring node only from the encrypted neighborlist of other nodes. Therefore, my formal analysis lead me to the following observation: in case

of link-state routing, all local neighborhood (routing) information that is needed by remote nodes to authenticate neighborhood relations must be transferred in an encrypted form.

Chapter 6

Secure Label-switching Routing in Wireless Sensor Networks

I show that the secure variant of TinyOS beaconing cannot defend against routing state pollution attacks either. In this chapter, I propose a novel secure routing protocol, called Secure-TinyLUNAR, for wireless sensor networks. I also adapt my model to secure label-switching routing by defining a general security objective for these routing protocols, and I formally prove that Secure-TinyLUNAR is secure in this model. Besides its provable security, another advantage of Secure-TinyLUNAR is that, similar to TinyLUNAR [Osipov, 2007], it uses label-switching routing, which results in a reduced addressing overhead during data packet forwarding.

Although there are some secure sensor network routing protocols in the literature, these are only applicable to specific sensor applications. Considering the variety of sensor applications, it is also clear that it is not possible to propose a unique secure routing protocol that fits for all applications as it has been already described in Section 2.2.3. An alternative solution could be to apply some secure ad hoc network routing protocol like [Zapata and Asokan, 2002; Sanzgiri *et al.*, 2002; Hu *et al.*, 2002]. However, these protocols are not primarily designed for low-powered sensor nodes, and the applied cryptographic primitives can result in extensive communication, processing and memory costs. Therefore, in this chapter, I design a novel secure routing protocol for wireless sensor networks, called Secure-TinyLUNAR, which takes into consideration the resource constraints of the wireless sensor nodes and uses Message Authentication Code (MAC) exclusively in the route discovery phase.

Secure-TinyLUNAR is the secure variant of TinyLUNAR [Osipov, 2007] which is a reactive routing protocol proposed for wireless sensor networks. Using the label-switching routing paradigm, TinyLUNAR has only one byte addressing overhead per packet in the data forwarding phase, which, considering the high communication costs in wireless environment, makes it an efficient routing scheme. Although TinyLUNAR has a slightly greater Random Access Memory (RAM) consumption than other reactive routing protocols like tinyAODV [Perkins and Royer, 1999], it uses considerably less Read Only Memory (ROM). These advantageous

properties become even more important if we take into account that Secure-TinyLUNAR uses some cryptographic primitives that also consume a significant amount of memory. Moreover, I show that due to the label switching mechanism intermediate nodes do not need to check the authenticity of the message origin that can save precious energy.

The outline of this chapter is as follows. In Section 6.1, I review the network and node assumptions of Secure-TinyLUNAR. I specify the protocol in Section 6.2, and I define the security objective of label-switching routing in Section 6.4. After discussing the tolerable imperfections incorporated by this security objective in Section 6.5, I prove that Secure-TinyLUNAR is indeed secure in this model in Section 6.6. I summarize the results in Section 6.7.

6.1 Assumptions

As in Section 3.5.1, I assume that there are adversarial nodes in the network that behave as it is described in Subsection 2.1.4.

I assume bidirectional links in the network, in case unidirectional links can occur, a mechanism is needed to eliminate such links. I do not consider attacks on the Medium Access Control protocol which is in use. I assume that links are unreliable (i.e., packets may be lost, corrupted, re-ordered, or duplicated in transmission).

Sensor network nodes do not show such high variety in terms of computational power than ad hoc network nodes. Typical sensor nodes like Crossbow Micaz motes [(MicaZ Datasheet), 2005] or Tmote Sky [(Tmote Sky Datasheet), 2006] motes have low performance Micro Controller Unit (MCU)s with constraint energy supply. Thus, instead of using resource demanding asymmetric cryptography, I use the more energy efficient symmetric key cryptography to provide security mechanisms in Secure-TinyLUNAR. Thus, I assume that each node is capable of generating Message Authentication Codes using pairwise shared keys (as it is described in [Ács and Buttyán, 2008b], one candidate for this is the TinySec package [Karlof *et al.*, 2004] which runs on top of TinyOS [Hill *et al.*, 2000]). According to this, it is assumed that each pair of nodes share a symmetric pairwise key in the network. Any symmetric key pre-distribution schemes proposed for wireless sensor networks (see [Çamtepe and Yener, 2005] for a good overview) can be employed for this purpose. Here, I note that Secure-TinyLUNAR as well as TinyLUNAR were mainly fuelled by distributed data storage applications like TinyPEDS [Girao *et al.*, 2007], where a sensor node needs to send data to another sensor node in the network field (i.e., point-to-point communication is required). Nevertheless, Secure-TinyLUNAR can also be used when sensor nodes only need to forward data to the base station. In that case, each sensor node needs to share a pairwise symmetric key only with the base station. Additionally, each node is assumed to be aware of its local (one-hop) neighborhood. In particular, similar to Section 3.5.1, I also assume that nodes can determine their local neighborhood by executing a (secure) neighbor discovery protocol. Finally, I also assume that nodes show low mobility during their operation.

6.2 Specification of Secure-TinyLUNAR

I only discuss the main operational differences with respect to the original (and insecure) TinyLUNAR [Osipov, 2007] protocol that has already been detailed in Section 2.1.2.

Route request: Let us denote the identifier of a neighboring node of node A by N_x^A, where x can have a value between 1 and the number of the neighboring nodes of A (e.g., if A has neighbors J, T, P, then a potential notation is $N_1^A = J$, $N_2^A = T$, $N_3^A = P$, and $1 \leq x \leq 3$).

When a node S wishes to send a message to destination D, it broadcasts the following route request message:

$$S \rightarrow * : (\mathsf{RREQ}, \mathsf{rnd}, S, D, addr_S, label^{\mathsf{In}}_{S \rightarrow S}, \mathsf{MAC}_{S,D})$$

where $\mathsf{rnd}, S, D, addr_S, label^{\mathsf{In}}_{S \rightarrow S}$ are the same as in the original TinyLUNAR protocol, and $\mathsf{MAC}_{S,D}$ is the message authentication code generated by S on the elements of the message excluding $addr_S$ and $label^{\mathsf{In}}_{S \rightarrow S}$ using the pairwise key shared with D. Upon the reception of this broadcast message, J checks whether S is a neighboring node. If so, node J unicasts the following message to each neighbor except the node who sent the request to J earlier (here, this is S):

$$\text{for all } x \text{ such that } N_x^J \neq S, J \rightarrow N_x^J : (\mathsf{RREQ}, \mathsf{rnd}, S, D, addr_J, label^{\mathsf{In}}_{J \rightarrow S}, \mathsf{MAC}_{S,D}, \mathsf{MAC}^{prv}_{J,N_x^J})$$

where $\mathsf{MAC}^{prv}_{J,N_x^J}$ is the previous-hop MAC generated on all elements of the message using the pairwise key shared between J and N_x^J. Each neighbor of S and all subsequent nodes receiving a request follow the same steps that J did (except that they update the previous-hop MAC). Finally, D receives a request message, let us assume, from node Z first.

During the propagation of a request, it is assumed that each node can send the unicast request message to its immediate neighbors in an atomic manner (i.e., the sender does not release the channel until all request messages are transmitted to each neighbor), and each neighboring node does not begin to forward the request until all neighbors of the sender receive that.

Route reply: Upon the reception of the request message, destination D verifies both $\mathsf{MAC}_{S,D}$ and $\mathsf{MAC}^{prv}_{Z,D}$. If the verifications are successful, D creates the following reply message and sends this directly to node Z:

$$D \rightarrow Z : (\mathsf{RREP}, \mathsf{rnd}, addr_D, label^{\mathsf{Out}}_{Z \rightarrow S}, label^{\mathsf{In}}_{D \rightarrow D}, \mathsf{MAC}_{D,S})$$

where rnd is the request id received in the corresponding route request message, and $\mathsf{MAC}_{D,S}$ is the message authentication code generated by D on the elements of the above message excluding $addr_D$, $label^{\mathsf{Out}}_{Z \rightarrow S}$, and $label^{\mathsf{In}}_{D \rightarrow D}$ using the pairwise key shared with S. Receiving this unicast message, Z first checks if D belongs to its neighborhood. If so, Z sends the message directly to node K, from which Z received the corresponding request message identified by rnd:

$$Z \rightarrow K : (\mathsf{RREP}, \mathsf{rnd}, addr_Z, label^{\mathsf{Out}}_{K \rightarrow S}, label^{\mathsf{In}}_{Z \rightarrow D}, \mathsf{MAC}_{D,S}, \mathsf{MAC}^{prv}_{Z,K})$$

Here, $\mathsf{MAC}^{prv}_{Z,K}$ is the previous-hop MAC generated by Z on the elements of the message including $addr_Z$, $label^{\mathsf{Out}}_{K\rightarrow S}$, and $label^{\mathsf{In}}_{Z\rightarrow D}$. Following the same rules, all intermediate nodes perform the same steps that Z did (except that each intermediate node updates the previous-hop MAC). Finally, the reply reaches the source S, which then, after verifying the previous-hop MAC and $\mathsf{MAC}_{D,S}$ in the reply message, can use the established route for data forwarding.

6.3 Computation and communication overhead of Secure-TinyLUNAR

Comparing to TinyLUNAR, Secure-TinyLUNAR requires the sender of a request message to perform two MAC generations. Furthermore, each node receiving a request must verify and generate one MAC. If we use a Cipher Block Chaining Message Authentication Code (CBC-MAC) construction with a common block cipher like Skipjack for MAC computation as proposed in [Karlof *et al.*, 2004], a MAC has a size of 64 bits. Therefore, there are 16 extra bytes in each request and reply packet. Note that this overhead is not constant at each hop in the request phase, as a node, compared to TinyLUNAR, does not broadcast request messages, rather it unicasts them to each neighboring node. The reason of this unusual design is that a request contains a pairwise MAC computed with the pairwise key shared between the sender and a particular neighbor, which is apparently not verifiable by other neighbors. If a node broadcast this request, a single broadcast message would be too long. As the packet size under TinyOS is suggested to be around 36 bytes [Hill *et al.*, 2000] and the number of neighbors of an ordinary sensor node is generally not fixed, most request messages would be fragmented. Moreover, broadcasting a request all receiver nodes would be required to receive all MACs that are not destined to them, which could yield significant overhead at every receiver node. This overhead is usually greater than the cost of sending the data part (node ids, network addresses, labels, source MAC, etc.) of a single request multiple times.

One might immediately ask why we do not use digital signatures [X9.63, 1999], μTesla [Perrig *et al.*, 2002], or local broadcast keys like in Localized Encryption and Authentication Protocol (LEAP) [Zhu *et al.*, 2003]? In case of local broadcast keys, when a common key is shared among the sender and all its neighbors, cannot guarantee neighbor authentication, as a neighboring adversarial node would be able to impersonate any honest neighbor using the shared key. Although μTesla does not have this drawback and it also uses efficient symmetric cryptography, it requires each receiver to maintain a hash chain. If route discoveries are invoked infrequently, which holds for most sensor networks due to their static nature, the verification of a particular broadcast key requires several evaluations of the employed hash function on average, which can result in significant computational overhead. Moreover, μTesla relies on a clock synchronization protocol which also incurs additional overhead on each node. Finally, digital signatures incur a substantial computation overhead. In particular, Public Key Cryptography (PKC) still falls behind the standard symmetric cryptography approaches in terms of computational performance; the verification of a digital signature is 3 orders of magnitude slower than MAC verification, while the signature generation is 4 orders of

magnitude slower.

In order to compare digital signatures with MACs in terms of energy cost regarding the route request phase of Secure-TinyLUNAR, we approximate the energy consumption of a single MICAz mote [(MicaZ Datasheet), 2005] in the route request phase. If we use the aforementioned MAC scheme, the previous-hop MAC is computed over 3 blocks (1 block is 8 bytes) which takes 1.14 ms [Karlof *et al.*, 2004] and consumes about 0.034 mWs [Piotrowski *et al.*, 2006]. If we assume that a node has at most 30 neighbors, all the computation cost is $30 \cdot 0.034 = 1.02$ mWs. If the radio transceiver operates at transmission speed of 250 kbit/s at 3 V supply voltage and the output power is set to 0 dBm (maximum power), then the power consumption is 0.209 μWs per bit for the transmission and 0.226 μWs per bit for the reception. Thus, as the size of a request packet is 33 bytes (including the header of the packet) under TinyOS [Hill *et al.*, 2000], the power consumption of the transmission is $30 \cdot 264 \cdot 0.000209 = 1.65528$ mWs. In addition, the reception of a request consumes $264 \cdot 0.000226 = 0.0596$ mWs. Therefore, all the communication overhead is about 1.715 mWs, and the communication and computation overhead together is about 2.735 mWs.

In contrast to this, using an optimized ECDSA [X9.63, 1999] [Piotrowski *et al.*, 2006] implementation with the shortest key-size (i.e., 160 bits) the signature generation and verification consumes 26.96 mWs and 53.42 mWs [Piotrowski *et al.*, 2006], resp. Thus, the total computation overhead of using digital signatures at one hop is more than 29 times larger than the total overhead (including computation and communication) of using MACs. Even if we used the more powerful TelosB motes [(Tmote Sky Datasheet), 2006], the total computation overhead of signatures would be 18.67 mWs which is about 7 times larger. Of course, sending multiple packets instead of a single one incurs extra costs in the medium access layer, but we believe that this extra cost still does not overcome the computation overhead of digital signatures. Moreover, generating and verifying an ECDSA-160 signature takes more than 2 seconds [Piotrowski *et al.*, 2006] which would also incur substantial network delay.

6.4 Security objective

According to the definition of the *cost function* in Subsection 2.2.2, $\mathcal{C} : V \to \mathbb{R}$ assigns the minimal delay of routing messages to each node in the network (i.e., the minimal delay that the particular node can incur during the travel of the message). We assume that $\mathcal{C}(v^*) = 0$ for all $v^* \in V^*$.

Before introducing the security objective function of label-switching routing in sensor networks, I introduce some definitions in order to ease its formalization.

Definition 7 (Anchor entry) *An anchor entry* $(v_{src}, v_{dest}, \mathsf{addr}_{nxt}, label^{\mathsf{Out}}_{v_{src} \to v_{dest}}, \mathsf{delay}_{v_{src}, v_{dest}})$ *is the representation of a routing entry at source* v_{src}*, where the destination node is identified by* v_{dest}*), the next-hop towards the destination has (local) address* addr_{nxt}*, the outgoing label of the source towards the destination is* $label^{\mathsf{Out}}_{v_{src} \to v_{dest}}$*, and the delay of the quickest path through* addr_{nxt} *to the destination is* $\mathsf{delay}_{v_{src}, v_{dest}}$*.*

Definition 8 (Intermediate entry) *An intermediate entry* $(v_{im}, \mathsf{addr}_{nxt}, label^{\mathsf{In}}_{v_{im}\to v_{dest}}, label^{\mathsf{Out}}_{v_{im}\to v_{dest}})$ *is the representation of a routing entry at an intermediate node* v_{im}*, where the next-hop towards the destination has (local) address* addr_{nxt}*, the incoming label and the outgoing label of* v_{im} *towards the destination are* $label^{\mathsf{In}}_{v_{im}\to v_{dest}}$ *and* $label^{\mathsf{Out}}_{v_{im}\to v_{dest}}$*, respectively.*

Definition 9 (Matching property) *A routing entry* r_1 *of node* v_i *matches a routing entry* r_2 *of node* v_j $(i \neq j)$*, if*

- *the outgoing label of* r_1 *equals to the incoming label of* r_2*,*
- *the next-hop address of* r_1 *is used by* v_j*.*

I recall that the state of the system is represented by the ensemble of all anchor and intermediate entries of all *honest* nodes.

Definition 10 (Correct state) *A state is correct with respect to configuration conf, if for every anchor entry* $r_0 = (v_{src}, v_{dest}, \mathsf{addr}_{nxt}, label^{\mathsf{Out}}_{v_{src}\to v_{dest}}, \mathsf{delay}_{v_{src},v_{dest}})$*, where* $v_{src}, v_{dest} \in V \setminus V^*$*, there exists a sequence of intermediate entries* $r_i = (v_{\ell_i}, \mathsf{addr}_{nxt}, label^{\mathsf{In}}_{v_{\ell_i}\to v_{dest}}, label^{\mathsf{Out}}_{v_{\ell_i}\to v_{dest}})$ $(1 \leq i \leq d)$ *of honest nodes such that*

- $v_{\ell_d} = v_{dest}$ *and* $label^{\mathsf{Out}}_{v_{dest}\to v_{dest}}$ *is an application identifier of* v_{dest}*,*
- $(v_{src}, v_{\ell_1}, \ldots, v_{\ell_{d-1}}, v_{dest})$ *is a workable path,*
- *let* $v_{\ell_0} = v_{src}$*,*
  - *if* $v_{\ell_{i-1}}$ *and* v_{ℓ_i} *are direct but not pseudo neighbors then* r_{i-1} *matches* r_i*,*
  - *if* $v_{\ell_{i-1}}$ *and* v_{ℓ_i} *are pseudo neighbors then either* r_{i-1} *matches* r_i*, or the next-hop address of* r_{i-1} *belongs to a neighboring adversarial node,*
- $\sum_{j=1}^{d-1} \mathcal{C}(v_{\ell_j}) \leq \mathsf{delay}_{v_{src},v_{dest}}$ *(i.e., the delay of the discovered route between* v_{src} *and* v_{dest} *is not greater than the delay recorded in the routing (anchor) entry of* v_{src}*)*

Let the security objective function $\mathcal{F}$ of secure label-switching routing return 0 for all pairs of system states and configurations that are incorrect, otherwise it returns 1. This function intends to distinguish "attacked" (incorrect) states from "non-attacked" (correct) states.

For the sake of simplicity, I assume that during the analysis the maximum lifetime of each entry is set to ∞.

6.5 Tolerable imperfections

Similarly to the secure protocols analysed so far, the adversary can also mount hidden channel attacks using the random message identifier in the control messages. However, similarly to ARAN in Section 4.3, for most real scenarios this attack is impractical, as by the time the last fragment of a particular MAC is successfully transferred, the corresponding control message probably becomes obsolete.

The third point in Definition 10 requires that if the next node on the route is a direct but not pseudo neighbor then r_{i-1} must match r_i, otherwise the next-hop address of r_i must belong to a neighboring adversarial node. The argument is similar to that of Definition 5 in Section 4.3. Namely, if $v_{\ell_{i-1}}, v_{\ell_i}$ are pseudo neighbors, then the adversary can modify the message before sending it to $v_{\ell_{i-1}}$. If the adversary relays the message without any modifications, r_{i-1} will match r_i, otherwise, she can only force $v_{\ell_{i-1}}$ to use the next-hop address of a neighboring adversarial node.

Finally, the last point in Definition 10, which is about the cost (delay) of the discovered route, relates to the fact that the adversary can always increase the delay of any message that passes it similarly to the case of ARAN in Section 4.3. In this way, it can make the cost of each route appear to be higher than it really is; we tolerate this in our model. On the other hand, this type of attack may be less attractive for the adversary, as increasing the delay of each route passing him can cause the source node to accept those routes that contain no adversarial nodes. If the adversary intends to fool the source node by making the cost of the discovered route appear lower than it is in reality (e.g., in order to increase the hostile traffic control by alluring the traffic), then the best that she can achieve is that she somehow reduces the delay of messages to zero at the adversarial nodes. However, as she cannot reduce the delay at the non-corrupted nodes, the appeared cost of the discovered route should always be greater than or equal to the sum of the cost of each node constituting this route.

6.6 Security proof

Theorem 7 *Secure-TinyLUNAR is a secure label-switching routing protocol for wireless sensor networks, if the MAC scheme is secure against chosen message attacks.*

Proof (sketch) I show that for any adversary $\mathcal{A}$ and any configuration *conf*, security objective function $\mathcal{F}$ equals to 0 only with probability that is a negligible function of κ. Equivalently, I show that the probability that for any adversary $\mathcal{A}$ and any configuration *conf* a system running Secure-TinyLUNAR encounters incorrect state is a negligible function of κ.

A system running Secure-TinyLUNAR encounters incorrect state in the cases as follows:

- Case 1: There exists an anchor entry $r_0 = (v_{src}, v_{dest}, \mathsf{addr}_{nxt}, label^{\mathsf{Out}}_{v_{src},v_{dest}}, \mathsf{delay}_{v_{src},v_{dest}})$, but there does not exist a workable path between v_{src} and v_{dest} with $label^{\mathsf{Out}}_{v_{\ell_d}\rightarrow v_{dest}}$ as an application identifier.
- Case 2: There exists an anchor entry $r_0 = (v_{src}, v_{dest}, \mathsf{addr}_{nxt}, label^{\mathsf{Out}}_{v_{src}\rightarrow v_{dest}}, \mathsf{delay}_{v_{src},v_{dest}})$ and at least one workable path between v_{src} and v_{dest}, but there does not exist a sequence of intermediate entries $r_i = (v_{\ell_i}, \mathsf{addr}_{nxt}, label^{\mathsf{In}}_{v_{\ell_i}\rightarrow v_{dest}}, label^{\mathsf{Out}}_{v_{\ell_i}\rightarrow v_{dest}})$ $(1 \leq i \leq d)$ for any existing workable paths such that
  - if $v_{\ell_{i-1}}, v_{\ell_i}$ are direct but not pseudo neighbors, then r_{i-1} matches r_i,
  - or, if $v_{\ell_{i-1}}, v_{\ell_i}$ are pseudo neighbors, then either r_{i-1} matches r_i or the next-hop address of r_{i-1} belongs to a neighboring adversarial node,

for all i.

- Case 3: There exists an anchor entry $r_0 = (v_{src}, v_{dest}, \mathsf{addr}_{nxt}, label^{\mathsf{Out}}_{v_{src}\to v_{dest}}, \mathsf{delay}_{v_{src},v_{dest}})$ and at least one sequence of intermediate entries $r_i = (v_{\ell_i}, \mathsf{addr}_{nxt}, label^{\mathsf{In}}_{v_{\ell_i}\to v_{dest}}, label^{\mathsf{Out}}_{v_{\ell_i}\to v_{dest}})$ $(1 \leq i \leq d)$ where $(v_{src}, v_{\ell_1}, \ldots, v_{\ell_{d-1}}, v_{dest})$ is a workable path, where either r_{i-1} matches r_i or the next-hop address of r_{i-1} belongs to a neighboring adversarial node for all i, but $\sum_{j=1}^{d-1} \mathcal{C}(v_{\ell_j}) > \mathsf{delay}_{v_{src},v_{dest}}$ for all such sequences.

We must prove that each of Case 1, 2 and 3 occurs only with a probability that is a negligible function of κ_1 and κ_2 which concludes the theorem.

Case 1 occurs, if v_{src} receives either a RREP or a RREQ message with a correct $\mathsf{MAC}_{v_{dest},v_{src}}$. Let us assume that the adversary $\mathcal{A}$ cannot forge $\mathsf{MAC}_{v_{dest},v_{src}}$. Thus, $\mathsf{MAC}_{v_{dest},v_{src}}$ can only be generated by v_{dest} implying that v_{dest} generated and sent a RREQ or RREP message with v_{src} as the destination, and $label^{\mathsf{Out}}_{v_{dest}\to v_{dest}}$ is an application identifier. Moreover, as $\mathsf{MAC}_{v_{dest},v_{src}}$ is received by v_{src}, there exists a sequence of nodes $(v_{s_0}, v_{s_1}, \ldots, v_{s_{k-1}}, v_{dest})$ such that $v_{s_{i-1}}, v_{s_i}$ are direct or pseudo neighbors for all $1 \leq i \leq k$, where $v_{src} = v_{s_0}$ and $v_{dest} = v_{s_k}$. This means that there is a workable path between v_{src} and v_{dest}. Therefore, Case 1 occurs only if the adversary successfully forges a MAC. However, the probability of this event is a negligible function of κ assuming that the adversary runs in polynomial time.

Case 2 occurs, if for *all* workable paths $(v_{\ell_0}, \ldots, v_{\ell_d})$ between v_{src} and v_{dest}, there is at least one pair $v_{\ell_{i-1}}, v_{\ell_i}$ of honest nodes such that

- Case 2a: $v_{\ell_{i-1}}, v_{\ell_i}$ are direct but not pseudo neighbors, however, r_{i-1} does not match r_i,
- Case 2b: or $v_{\ell_{i-1}}, v_{\ell_i}$ are pseudo neighbors, however, r_{i-1} does not match r_i and the next-hop address of r_{i-1} does not belong to a neighboring adversarial node either.

Let us assume that $\mathcal{A}$ cannot forge any MACs. As v_{src} has anchor entry r_0, v_{src} receives either a RREP or a RREQ message with a correct $\mathsf{MAC}_{v_{dest},v_{src}}$. Thus, based on Case 1, there exists a workable path $(v_{\ell_0}, \ldots, v_{\ell_d})$ between v_{src} and v_{dest} along which the request (or reply) message, denoted by msg, is received by v_{src}. In case 2a, according to our assumption, there exists i such that $v_{\ell_{i-1}}, v_{\ell_i}$ do not have matching entries, however, they are direct but not pseudo neighbors. As $\mathsf{MAC}^{prv}_{v_{\ell_i},v_{\ell_{i-1}}}$ can only be generated by v_{ℓ_i}, $v_{\ell_{i-1}}$ received an msg' message $(msg' \neq msg)$ with previous-hop MAC $\mathsf{MAC}^{prv}_{v_x,v_{\ell_{i-1}}}$, where $\mathsf{MAC}^{prv}_{v_x,v_{\ell_{i-1}}} \neq \mathsf{MAC}^{prv}_{v_{\ell_i},v_{\ell_{i-1}}}$. Since $\mathsf{MAC}_{v_{dest},v_{src}}$ travelled through workable path $(v_{\ell_0}, \ldots, v_{\ell_d})$, v_x is an adversarial node and the adversary obtained $\mathsf{MAC}_{v_{dest},v_{src}}$ from v_{ℓ_i}. Therefore, both v_{ℓ_i} and $v_{\ell_{i-1}}$ have an adversarial neighbor, which means that they are pseudo neighbors. However, this contradicts to our assumption that v_{ℓ_i} and $v_{\ell_{i-1}}$ cannot be pseudo neighbors. In case 2b, if the next-hop address of r_{i-1} cannot be an adversarial one, then $\mathsf{MAC}^{prv}_{v_x,v_{\ell_{i-1}}}$ can only be generated by an honest node v'. As $\mathsf{MAC}_{v_{dest},v_{src}}$ travelled through a workable path between v_{src} and v_{dest}, $v' = v_{\ell_i}$ which means that $r_{\ell_{i-1}}$ must match r_{ℓ_i}. However, if $r_{\ell_{i-1}}$ does not match r_{ℓ_i}, then the next-hop address of r_{i-1} must be an adversarial one based on case 2a. Consequently, Case 2 occurs

only if the adversary successfully forges a MAC. However, the probability of this event is a negligible function of κ assuming that the adversary runs in polynomial time.

Finally, in Case 3, $\mathsf{delay}_{v_{src},v_{dest}}$ denotes the delay of the travel of $\mathsf{MAC}_{v_{dest},v_{src}}$ from its originator to v_{src} (either as a part of a RREQ or a RREP control message). Let us assume that $\mathsf{MAC}_{v_{dest},v_{src}}$ cannot be forged by the adversary $\mathcal{A}$. Based on Case 1 and Case 2, $\mathsf{MAC}_{v_{dest},v_{src}}$ is received on a workable path $(v_{\ell_0},\ldots,v_{\ell_d})$, and hence, $\sum_{j=1}^{d-1}\mathcal{C}(v_{\ell_j}) > \mathsf{delay}_{v_{src},v_{dest}}$ is only possible, if the adversary could speed up the transmission on links or the processing at the non-corrupted nodes. Consequently, Case 3 occurs only if the adversary successfully forges a MAC. However, the probability of this event is a negligible function of κ assuming that the adversary runs in polynomial time. ∎

6.7 Summary

This chapter discussed the security of label-switching routing in wireless sensor network. I designed a novel secure label-switching protocol, called Secure-TinyLUNAR, which is based on TinyLUNAR described in Subsection 2.1.2. After defining the security objective of label-switching routing in sensor networks, I proved that Secure-TinyLUNAR is secure regarding this security objective.

Apart from being provably secure, Secure-TinyLUNAR also has some nice properties. Particularly, intermediate nodes do not need to check the authenticity of the message origin, which means that the source does not need to use expensive global broadcast authentication methods based on asymmetric cryptography. Instead, Secure-TinyLUNAR uses pairwise MACs based on the more-energy conserving symmetric key cryptography for previous-hop (neighbor) and message origin authentication. Secure-TinyLUNAR provides the following security guarantees:

- Each node generates a MAC per neighbor on the request message, and unicasts the request along with the respective MAC to each neighbor. Although this previous-hop MAC is updated at each hop, the communication and computation costs depend on the number of the neighbors. A reply message also contains a previous-hop MAC that is updated at each hop, but it is always sent to one neighbor which results in a constant overhead for all intermediate hops.

- The source and destination nodes attach a MAC to each message. As this MAC is generated by using the pairwise shared key of the source and destination nodes, intermediate nodes do not need to verify this MAC saving some resources. Nevertheless, the protocol is provably secure in my model, even if these MACs are not verified by intermediate nodes. On the other hand, without verifying the authenticity of the origin, the protocol remains vulnerable to DoS attacks. However, note that employing a one-way hash chain to authenticate the source is sufficient in this case. Moreover, hash chains have much less computational and communication overhead than other broadcast authentication schemes [Ács and Buttyán, 2008b].

Chapter 7

Application of New Results

In the last decade, several security protocols were proposed without any rigorous security proofs. However, informal reasoning is not sufficient to provide a real assurance of security. This is also proved by the fact that many of these protocols were flawed after their first proposals. Thus, there is a continuous research effort on the development of sound formal models and precise proof techniques which could assist the design of cryptographic protocols.

The lesson learnt, hence, my work attempts to assist the validation of secure routing protocols proposed primarly for wireless ad hoc and sensor networks. As these protocols are very subtle due to the wireless nature of communications, I believe that the formal validation of these security protocols is essential. Although the validation process has not been automated yet, a protocol designer can easily prove the security of any multi-hop routing protocols in wireless context by using the given proof technique. If the protocol is insecure in my model, the designer can find an attack by investigating where the proof fails.

Another benefit of my work is that I have identified several design principles of secure routing in wireless ad-hoc and sensor networks by proving the security of various routing protocols in my model. For instance, these include the requirements of route reply authentication in dynamic source routing, the per-hop authentication in dynamic distance vector routing, or the encryption of local topology information in link-state routing. Even if a protocol designer did not use any formal methods to validate a newly proposed secure routing protocol, following these principles he could avoid many potential vulnerabilities that are anyway difficult to identify by using informal reasoning exclusively.

Some of my results were used within an European research project called UbiSec&Sens (Ubiquitous Sensing and Security in the European Homeland, http://www.ist-ubisecsens.org/) between 2005 and 2008. The project was an IST STReP and received research funding from the European Community's Sixth Framework Programme. The primary objective of UbiSec&Sens was to provide a security and reliability architecture for medium and large-scale wireless sensor networks acting in volatile environments. In particular, it provided a complete toolbox of security and reliability aware components for sensor network application development. UbiSec&Sens' work is focused on the intersection of security, routing and in-network processing to design and develop efficient and effective security solutions and to offer effective means for persistent and encrypted data storage for

distributed (and tiny) data base approaches. The solutions are prototyped and validated in the representative wireless sensor application scenarios of agriculture, road services and homeland security.

In Ubisec&Sens, I developed the formal framework described in Section 2.2 and applied it in designing a provably secure routing protocol, called Secure-TinyLUNAR (described in Section 6.2), for wireless sensor networks. Based on the available nesC source code of TinyLUNAR under TinyOS 2.x [TinyOS 2.x, 2007], I implemented Secure-TinyLUNAR by extending TinyLUNAR with the security mechanisms described in Section 6.2. As a part of this work, I compared the performance of TinyLUNAR, which is an extensively used routing protocol in UbiSec&Sens, and Secure-TinyLUNAR. For performance evaluation, I used TOSSIM [Levis *et al.*, 2003] which is a packet-level simulator for TinyOS 2.x. For energy measurement, I extended TOSSIM with PowerTOSSIM 2 [Shnayder *et al.*, 2004] that can be downloaded from the contrib part of the TinyOS 2.x distribution [TinyOS Alliance, 2007]. I concluded that while the consumed computational energy of Secure-TinyLUNAR is comparable to that of TinyLUNAR, the employed security mechanisms introduce extra communicational costs and network delay. In particular, the computation overhead of TinyLUNAR and Secure-TinyLUNAR is almost the same (to be more precise, Secure-TinyLUNAR consumes about 3% more computational energy than TinyLUNAR). This proves that the employment of pairwise MACs causes minimal overhead in terms of computation. However, the communicational overhead is about 5 times larger on average. One obvious reason is that applying MACs results in 88 bytes extra overhead per packet on average (one source MAC and 10-12 previous-hop MACs). Second, as we have not integrated Secure-TinyLUNAR with MAC (Medium Access Layer) layer, our "pseudo" broadcast mechanism used to propagate RREQ messages causes increased overhead in the MAC layer.

Another European research project focusing on the security of critical infrastructures is called WSAN4CIP (Wireless Sensor and Actuator Networks for Critical Infrastructure Protection, http://www.wsan4cip.eu/), which is started at the beginning of 2009 and lasts for 3 years. This project is also an IST STReP and receives research funding from the European Community's Seventh Framework Programme. The project goals are to enhance the reliability of critical infrastructures by providing surveillance data for the management of the critical infrastructure using wireless sensor and actuator networks (WSANs), and to increase the dependability of critical infrastructures security by providing self-healing and dependability modules for WSANs. The project also aims at providing appropriate tool support, and demonstrating the feasibility of this approach using energy generation and distribution as a representative of critical infrastructures. As such, the project will develop models for the reliability and security analysis of routing protocols for WSANs, which will be used to analyse various routing protocols both in terms of prevention of and reaction to attacks (the latter means attack detection and efficient recovery by reorganization of the overlay network topology). Here, prevention naturally includes the formal security validation of the route discovery phase of routing protocols which is the focus of my dissertation.

Chapter 8

Conclusions

The security of routing protocols is a fundamental issue in multi-hop wireless networks. This dissertation addressed the problem of secure routing in wireless ad hoc and sensor networks.

First, I gave an up-to-date picture of the security problems of routing protocols in ad hoc and sensor networks. In particular, in the first part of the dissertation, I described the ad hoc and sensor routing specific adversary model with her primary objectives, and I also listed some possible attack methods used by this adversary to achieve her objectives. I showed that many existing routing protocols are insecure against routing state pollution attacks, when the adversary cause honest nodes to store incorrect routing entries. I argued that routing state pollution attacks can be very subtle, and difficult to discover. As current models either use inappropriate assumptions or they are not general enough to analyse routing security in multi-hop wireless environment, I proposed a novel formal framework for this purpose. The model considers the wireless and multi-hop nature of these networks, the variety of routing security objectives, and the wireless specific adversary model. An insider adversary is assumed, which means that the adversary can have corrupted nodes, and thus, disregarding the protocol rules, she can arbitrarily manipulate routing messages at these nodes. By defining the security objective function, the model can be tailored to different families of routing protocols, where each member becomes comparable in terms of security.

I demonstrated the usefulness of my framework by several examples. First, I designed a novel source routing protocol called endairA, and I proved that it is secure. On the contrary, Ariadne is proved to be insecure, which means that my model is capable of distinguishing source routing protocols in terms of security. A consequence of the analysis is that a secure source routing protocol should always authenticate route reply messages in order to prevent routing state pollution attacks. Another attractive property of endairA is that it is more efficient than Ariadne, as only reply messages are required to be signed which are propagated back to the source on a single route. In turn, request messages flood the whole network causing substantially more overall energy consumption.

Second, I adapted my model to distance vector routing, and I proved that ARAN is secure in this model, whereas SAODV is not. I showed that, likewise SAODV, without neighbor (previous-hop) authentication a distance vector routing protocol cannot be secure against routing state pollution attacks.

Third, I also analysed the security of centralized link-state routing protocols in wireless sensor networks. I adapted my model to centralized link-state routing, and I proved that INSENS is secure in that model. I showed that a secure link-state routing protocol should always encrypt local topological information in order to prevent node impersonation, and eventually, routing state pollution attacks.

Finally, I analysed the security of label-switching routing in wireless sensor networks. Based on TinyLUNAR, I designed a novel secure label-switching routing protocol called Secure-TinyLUNAR. Instead of expensive digital signatures, this protocol employs only MACs to authenticate routing messages. Due to the label-switching paradigm, intermediate nodes do not need to verify the origin of the routing messages, just the authenticity of the neighbor who sent the message. Hence, costly global broadcast authentication methods can be avoided to save energy on sensor nodes. I adapted my model to label-switching routing, and I proved that Secure-TinyLUNAR is indeed secure in that model.

There is an on-going research effort on designing new secure routing protocols for wireless ad hoc and sensor networks. The message of this dissertation is that designing such routing protocols should always be assisted by formal verification, as attacks can be very subtle, and informal reasoning is insufficient to discover all of them. Although my model itself cannot be used to discover such attacks explicitly, but one can reveal flaws by inspecting where the proof fails.

This dissertation did not address the automation of proofs and the performance evaluation of provably secure routing protocols. This is left for future work. In addition, the adversary has been assumed to be non-adaptive (i.e., she cannot replay routing messages from earlier protocol runs) and only routing state pollution attacks are considered. Are endairA, ARAN, INSENS or Secure-TinyLUNAR still secure against a stronger adversary? If they are, then how this extended adversary model changes the complexity of proofs? Finally, the considered security objectives have disregarded which nodes constitute a route. However, a secure routing protocol should also avoid using adversarial nodes to forward data. In these cases, such techniques are needed that allow honest nodes to detect misbehaving nodes in the network. It is yet unclear, what the security objectives are in these cases, and how they can be modelled in a formal way. A related problem is that honest nodes are assumed not to use compromised identifiers in the network, which is a restriction of my framework. In Appendix A, I proved that endairA is secure assuming that some honest nodes use compromised identifiers.

Appendix A

The Security Proof of endairA with Compromised Nodes

In some practical scenarios, the adversary can capture honest nodes in order to gain their cryptographic keys, but then releases them. As a result, these *compromised nodes* behave further as honest nodes, but their secret keys are compromised and can be used by the adversary in the network. In contrast to adversarial nodes, which the adversary occupies and can eavesdrop on all incident communication, the adversary cannot control compromised nodes. Hence, this modified adversary model is weaker than the one which has been assumed throughout the dissertation. Thus, intuitively, all protocols that are proved to be secure regarding the stronger adversary should remain secure in this modified model. As an example, in the following, I formally prove that endairA remains secure if compromised nodes are assumed.

A.1 Adversary and network model

Let L denote the set of identifiers that are used by honest nodes. Similarly to Sections 2.2.2 and 2.2.1, I assume that the adversary has (authenticated) identifier(s) in the network, denoted by L^* that can be used by all adversarial nodes (i.e., $\mathcal{L}(v_{k+j}) = L^*$ for all $1 \leq j \leq m$). For every honest node v_i $(0 \leq i \leq k)$, $\mathcal{L}(v_i)$ is a singleton, *but* $L \cap L^* \neq \oslash$ (i.e., there are nodes which behave as honest nodes, but they use compromised identifiers).

We call a sequence of node identifiers a *compromised route*, if the first and the last identifier is non-compromised, but each intermediate identifier in the sequence is compromised.

A.2 Security proof

Definition 11 (Plausible compromised route) *A compromised route* $(z_1, z_2, \ldots, z_s)$ *is a plausible with respect to configuration* $conf$*, if there exists a workable path*[1] $(v_{\ell_1}, v_{\ell_2}, \ldots, v_{\ell_t})$ *such that for all* $1 \leq i \leq t-1$*,*

[1] Recall that a workable path is a sequence of honest nodes, where consecutive nodes are direct or pseudo neighbors.

- $\mathcal{L}(v_{\ell_i}) \in \{z_1, \ldots, z_s\}$, *where* $\mathcal{L}(v_{\ell_1}) = z_1$ *and* $\mathcal{L}(v_{\ell_t}) = z_s$,
- *for all* $1 \leq i \leq t-1$, $\mathcal{L}(v_{\ell_i}) = z_x$ $(1 \leq x \leq s-1)$ *implies that* $\mathcal{L}(v_{\ell_{i+1}}) = z_{x+y}$ *for some* $1 \leq y \leq s-1$. *Moreover, if* $y \geq 2$, *then* v_{ℓ_i} *and* $v_{\ell_{i+1}}$ *are pseudo neighbors.*

In order to prove the security of endairA, I first prove that endairA returns a route which contains a non-plausible *compromised* route only with negligible probability. This security objective function is denoted by $\mathcal{F}^{\mathsf{c}}$, and it is 0, if the route returned by endairA contains a non-plausible compromised route. Otherwise, $\mathcal{F}^{\mathsf{c}}$ is 1. Based on this proof, I prove that endairA returns a non-plausible route with negligible probability which is the same security objective that is detailed in Section 3.2 and it is further denoted by $\mathcal{F}^{\mathsf{p}}$.

Lemma 1 *endairA is secure with respect to* $\mathcal{F}^{\mathsf{c}}$, *if the signature-scheme is secure against chosen message attack.*

Proof (sketch) Let us suppose that a route containing a non-plausible compromised route $R = (z_1, z_2, \ldots, z_s)$ is returned to a non-adversarial machine z_{ini} in $sys_{conf,\mathcal{A}}$.

The reasons that R is a non-plausible compromised route with respect to *conf* can be as follows.

- *Case 1:* there does not exist any workable path between v_{ℓ_1} and v_{ℓ_t} such that the identifiers of all nodes on the path are in R,
- *Case 2:* Case 1 does not hold, but for all workable paths between v_{ℓ_1} and v_{ℓ_t}, there exists i such that $\mathcal{L}(v_{\ell_{i+1}}) \notin \{z_{x+1}, \ldots, z_{s-1}\}$ $(1 \leq x \leq s-1)$,
- *Case 3:* Case 1 and 2 do not hold, but for all workable paths between v_{ℓ_1} and v_{ℓ_t}, there exists i such that $\mathcal{L}(v_{\ell_{i+1}}) \in \{z_{x+2}, \ldots, z_{s-1}\}$ $(1 \leq x \leq s-1)$, but v_{ℓ_i} and $v_{\ell_{i+1}}$ are not pseudo neighbors.

Let us assume that the adversary cannot forge the signature of any honest node which has non-compromised identifier.

In Case 1, if the signature of z_s in the reply is not forged, then the very fact that v_{ℓ_1} received the reply proves that there is a workable path between v_{ℓ_1} and a node that uses z_s. Moreover, an honest node only forwards the reply if its identifier is in R, which means that the identifiers of all honest nodes that the reply traversed must be in R. Hence, Case 1 is possible only if the signature of z_s is forged.

In Case 2, let us denote the *shortest* workable path between v_{ℓ_1} and v_{ℓ_t} by $w = (v_{\ell_1}, \ldots, v_{\ell_i}, v_{\ell_{i+1}}, \ldots, v_{\ell_t})$, where $\mathcal{L}(v_{\ell_j}) \in R$ for all $1 \leq j \leq t$. According to our assumption, there exists i such that $\mathcal{L}(v_{\ell_{i+1}}) \notin \{z_{x+1}, \ldots, z_{s-1}\}$, where $z_x = \mathcal{L}(v_{\ell_i})$. This means that $v_{\ell_{i+1}}$ received the reply before v_{ℓ_i}, and $z_y = \mathcal{L}(v_{\ell_{i+1}})$ precedes z_x in R. In that case, the reply traversed at least one adversarial node between $v_{\ell_{i+1}}$ and v_{ℓ_t}, and thus, there must be at least one honest node between $v_{\ell_{i+1}}$ and v_{ℓ_t}, denoted by v_{ℓ_r}, which has an adversarial neighbor. Otherwise, the reply could not reach $v_{\ell_{i+1}}$, as an honest between $v_{\ell_{i+1}}$ and v_{ℓ_t} would detect the wrong order of identifiers in R. Similarly, there also must be at least one honest node

between v_{ℓ_1} and v_{ℓ_i}, denoted by $v_{\ell_{r'}}$, which has an adversarial neighbor. Therefore, there is a workable path

$$w' = (v_{\ell_1}, \ldots, v_{\ell_{r'}}, v_{\ell_r}, \ldots, v_{\ell_t})$$

which does not include v_{ℓ_i} and $v_{\ell_{i+1}}$. Hence, w' is a shorter workable path between v_{ℓ_1} and v_{ℓ_t} than w, which is a contradiction.

In Case 3, let us denote the path between v_{ℓ_1} and v_{ℓ_t} which the reply traversed by $w = (v_{\ell_1}, \ldots, v_{\ell_i}, v_{\ell_{i+1}}, \ldots, v_{\ell_t})$. v_{ℓ_i} could not receive the reply from $v_{\ell_{i+1}}$ directly, because $v_{\ell_{i+1}}$ never sends the reply to v_{ℓ_i} if $\mathcal{L}(v_{\ell_i})$ and $\mathcal{L}(v_{\ell_{i+1}})$ are not consecutive identifiers in R. By assumption, either $v_{\ell_{i+1}}$ or v_{ℓ_i} does not have any adversarial neighbors. In both cases, there must be a non-adversarial node v' which forwaded the reply between v_{ℓ_i} and $v_{\ell_{i+1}}$, which means that the reply traversed path

$$w' = (v_{\ell_1}, \ldots, v_{\ell_i}, v', v_{\ell_{i+1}}, \ldots, v_{\ell_t})$$

Clearly, $w' \neq w$, which is a contradiction. ∎

Theorem 8 *endairA is a secure source routing protocol for wireless ad hoc networks with respect to $\mathcal{F}^{\mathsf{p}}$, if the signature scheme is secure against chosen message attacks.*

Proof (sketch) Let us suppose that the following route reply is received by a non-adversarial machine z_{ini} in $sys_{conf,\mathcal{A}}$:

$$msg = (\mathsf{rrep},\ z_{ini},\ z_{tar},\ (z_1, \ldots, z_p),\ (sig_{z_{tar}}, sig_{z_p}, \ldots, sig_{z_1}))$$

Let us suppose that msg is accepted by z_{ini} (i.e., it passes all the verifications required by endairA at z_{ini}), and the route $(z_{ini}, z_1, \ldots, z_p, z_{tar})$ received in msg is non-plausible with respect to $conf$.

Similar to the proof of Theorem 3, all sequences of identifiers can unambiguously be partitioned such that each non-compromised identifier form a single partition and all consecutive compromised identifiers (between two non-compromised identifiers) also form a single partition. Let $P_1, P_2, \ldots, P_k$ be such a unique partitioning of the route $(z_{ini}, z_1, \ldots, z_p, z_{tar})$. As $(z_{ini}, z_1, \ldots, z_p, z_{tar})$ is a non-plausible route, according to Definition 3, at least one of the following two statements holds:

- *Case 1:* There exist two partitions $P_i = \{z_j\}$ and $P_{i+1} = \{z_{j+1}\}$ such that both z_j and z_{j+1} are non-compromised identifiers, and there does not exist honest nodes v_x, v_y ($v_x, v_y \in V \setminus V^*$) such that $\mathcal{L}(v_x) = z_j$ and $\mathcal{L}(v_y) = z_{j+1}$ and v_x, v_y are direct or pseudo neighbors.

- *Case 2:* There exist three partitions $P_i = \{z_j\}$, $P_{i+1} = \{z_{j+1}, \ldots, z_{j+q}\}$, and $P_{i+2} = \{z_{j+q+1}\}$ such that z_j and z_{j+q+1} are non-compromised and $z_{j+1}, \ldots, z_{j+q}$ are compromised identifiers, and $(z_j, z_{j+1}, \ldots, z_{j+q}, z_{j+q+1})$ is a non-plausible compromised route.

However, based on Theorem 3 and Lemma 1, these cases can only occur if the adversary successfully forges the digital signature of a non-adversarial machine. ∎

Acronyms

SAODV Secure Ad hoc On Demand Distance Vector routing

ARAN Authenticated Routing for Ad hoc Networks

INSENS Intrusion-Tolerant Routing in Wireless Sensor Networks

AODV Ad hoc On Demand Distance Vector routing

TinyLUNAR Tiny Lightweight Underlay Adhoc Routing for Wireless Sensor Networks

Secure-TinyLUNAR Secure Tiny Lightweight Underlay Adhoc Routing for Wireless Sensor Networks

DSR Dynamic Source Routing

TinyOS Tiny Operating System for wireless embedded sensor networks

WRP Wireless Routing Protocol

DSDV Destination Sequence Distance Vector Routing protocol

DREAM Distance Routing Effect Algorithm for Mobility

OLSR Optimized Link State Routing

ZRP Zone Routing Protocol

ZHLS Zone-based Hierarchical Link State routing

HARP Hybrid Ad hoc Routing Protocol

RREP Route Reply

RREQ Route Request

LEACH Low Energy Adaptive Clustering Hierarchy

TEEN Threshold sensitive Energy Efficient sensor Network protocol

APTEEN Adaptive Periodic Threshold sensitive Energy Efficient sensor Network protocol

GOAFR Greedy Other Adaptive Face Routing

GPSR Greedy Perimeter Stateless Routing

GEAR Geographic Energy Aware Routing

EAR Energy Aware Routing

PDA Personal Digital Assistants

GG Gabriel Graph

RNG Relative Neighborhood Graph

CLDP Crossing Link Detection Protocol

LCLR Lazy Cross Link Removal

MAC Message Authentication Code

CBC-MAC Cipher Block Chaining Message Authentication Code

LEAP Localized Encryption and Authentication Protocol

Tesla Timed, Efficient, Streaming, Loss-tolerant authentication protocol

RSA Rivest Shamir Adleman asymmetric-key based cryptographic algorithms

ECDSA Elliptic Curve Digital Signature Algorithm

PKI Public Key Infrastructure

PKC Public Key Cryptography

DoS Denial-of-Service

CPAL-ES Cryptographic Protocol Analysis Language Evaluation System

SRP Secure Routing Protocol

BAN Burrows-Abadi-Needham logic

DSA Digital Signature Algorithm

ROM Read Only Memory

RAM Random Access Memory

MCU Micro Controller Unit

SRDP Secure Route Discovery Protocol

List of publications

[Ács and Buttyán, 2005a] G. Ács and L. Buttyán. Ad hoc útvonalválasztó protokollok bizonyított biztonsága. *Híradástechnika*, 60(3):41–45, March 2005.

[Ács and Buttyán, 2005b] G. Ács and L. Buttyán. Provable security for ad hoc routing protocols. *Híradástechnika (English Edition)*, 60(6):34–38, June 2005.

[Ács and Buttyán, 2006] G. Ács and L. Buttyán. Útvonalválasztó protokollok vezeték nélküli szenzorhálózatokban. *Híradástechnika*, 61(12):3–12, December 2006.

[Ács and Buttyán, 2007] G. Ács and L. Buttyán. A taxonomy of routing protocols for wireless sensor networks. *Híradástechnika (English Edition)*, 62(1):32–41, January 2007.

[Ács and Buttyán, 2008a] G. Ács and L. Buttyán. Designing a secure label-switching routing protocol for wireless sensor networks. *To appear in Periodica Polytechnica (http://www.pp.bme.hu/)*, December 2008.

[Ács and Buttyán, 2008b] G. Ács and L. Buttyán. *Secure Routing in Wireless Sensor Networks.* Book chapter in Wireless Sensor Network Security (Cryptology and Information Security Series), Eds. J. Lopez and J. Zhou, ISBN: 978-1-58603-813-7, IOS Press, 2008.

[Ács *et al.*, 2005a] G. Ács, L. Buttyán, and I. Vajda. Provable security of on-demand distance vector routing in wireless ad hoc networks. In *Proceedings of the Second European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS 2005)*, pages 113–127, Visegrád, Hungary, July 2005.

[Ács *et al.*, 2005b] G. Ács, L. Buttyán, and I. Vajda. Provably secure on-demand source routing in mobile ad hoc networks. *Technical Report, CrySyS Lab, Budapest University of Technology and Economics. Also available at http://eprint.iacr.org/ under report number 2004/159.*, April 2005.

[Ács *et al.*, 2006a] G. Ács, L. Buttyán, and I. Vajda. Modelling adversaries and security objectives for routing protocols in wireless sensor networks. In *Proceedings of the Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2006)*, pages 49–58, Alexandria, VA, USA, October 2006.

[Ács *et al.*, 2006b] G. Ács, L. Buttyán, and I. Vajda. Provably secure on-demand source routing in mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(11):1533–1544, 2006.

[Ács *et al.*, 2007] G. Ács, L. Buttyán, and I. Vajda. The security proof of a link-state routing protocol for wireless sensor networks. In *Proceedings of the 3rd IEEE Workshop on Wireless and Sensor Networks Security (WSNS 2007)*, pages 1–6, Pisa, Italy, October 2007.

Bibliography

[Al-Karaki and Kamal, 2004] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11:6–28, 2004.

[Backes and Pfitzmann, 2004] M. Backes and B. Pfitzmann. A cryptographically sound security proof of the needham-schroeder-lowe public-key protocol. *IEEE Journal on Selected Areas of Computing (JSAC)*, 22(10):2075–2086, 2004.

[Basagnia *et al.*, 1998] S. Basagnia, I. Chlamtac, V. Syrotiuk, and B. WoodWard. A distance routing effect algorithm for mobility (DREAM). In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking (Mobicom)*, 1998.

[Bellare *et al.*, 1998] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *Proceedings of the ACM Symposium on the Theory of Computing*, 1998.

[Bharghavan *et al.*, 1994] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. Macaw: A media access protocol for wireless lans. In *In Proceedings of the SIG-COMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 212–225, 1994.

[Boneh *et al.*, 2003] D. Boneh, C. Gentry, H. Shacham, and B. Lynn. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology (Eurocrypt 2003)*, 2003.

[Broch *et al.*, 1998] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the Fourth Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 85–97, 1998.

[Bryan *et al.*, 2005] P. Bryan, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2005.

[Burmester and de Medeiros, 2009] M. Burmester and B. de Medeiros. Towards provable security for route discovery protocols in mobile ad hoc networks. In *IEEE Transacions on Mobile Computing (to appear, available on Cryptology ePrint Archive, under report number 2007/324)*, 2009.

[Burrows and Needham, 1990] M. Burrows and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8:18–36, 1990.

[Burrows *et al.*, 1990] M. Burrows, M. Abadi, and R. Needham. Rejoinder to nessett. *SIGOPS Oper. Syst. Rev.*, 24(2):39–40, 1990.

[Buttyán and Vajda, 2004] L. Buttyán and I. Vajda. Towards provable security for ad hoc routing protocols. In *Proceedings of the 2nd ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, October 2004.

[Committee, 1997] IEEE Computer Society LAN MAN Standards Committee. Medium access control (mac) and physical layer (phy) specifications,ieee std 802.11-1997. Technical report, The Institute of Electrical and Electronics Engineers, 1997.

[Deng *et al.*, 2002] J. Deng, R. Han, and S. Mishra. INSENS: Intrusion-tolerant routing in wireless sensor sensor networks. Technical Report CU-CS-939-02, Department of Computer Science, University of Colorado, 2002.

[Deng *et al.*, 2003] J. Deng, R. Han, and S. Mishra. A performance evaluation of intrusion-tolerant routing in wireless sensor networks. In *Proceedings of the IEEE Workshop on Information Processing in Sensor Networks (IPSN)*, pages 349–364, 2003.

[Dolev and Yao, 1981] D. Dolev and A. C. Yao. On the security of public key protocols. In *Proceedings of the IEEE 22nd Annual Symposium on Foundations of Computer Science*, pages 350–357, 1981.

[Douceur, 2002] J. R. Douceur. The sybil attack. In *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[dsa, 1994] FIPS 186. digital signature standard, 1994.

[Gabriel and Sokal, 1969] K. Gabriel and R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[Girao *et al.*, 2007] J. Girao, D. Westhoff, E. Mykletun, and T. Araki. Tinypeds: Tiny persistent encrypted data storage in asynchronous wireless sensor networks. *Elsevier Ad Hoc Networks*, 5(7):1073–1089, September 2007.

[Guttman, 2001] J. D. Guttman. Security goals: Packet trajectories and strand spaces. In *Foundations of Security Analysis and Design, volume 2171 of LNCS*, pages 197–261. Springer Verlag, 2001.

[Haas and Pearlman, 1998] Z.J. Haas and M.R. Pearlman. The zone routing protocol (zrp) for ad hoc networks. In *IETF Internet draft*, 1998.

[Heinzelman *et al.*, 2000] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS)*, 2000.

[Hill *et al.*, 2000] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2000.

[Hu and Perrig, 2004] Y.-C. Hu and A. Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy Magazine*, 2(3):28–39, 2004.

[Hu *et al.*, 2002] Y.-C. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of ACM Conference on Mobile Computing and Networking (Mobicom)*, 2002.

[Hu *et al.*, 2005] Y.-C. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. *Wireless Networks Journal*, 11(1), 2005.

[Hu, 2003] Yih-chun Hu. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, pages 30–40, 2003.

[Hubaux *et al.*, 2001] J. P. Hubaux, L. Buttyán, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*, Long Beach, CA, USA, October 2001.

[(IEEE), 2003] IEEE Standard for Information technology (IEEE). Telecommunications and information exchange between systems—local and metropolitan area networks—specific requirements. part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (lr-wpans), 2003.

[Intanagonwiwata *et al.*, 2000] C. Intanagonwiwata, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 55–67, 2000.

[Jacquet *et al.*, 2001] P. Jacquet, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *Proceedings of the IEEE International Multitopic Conference (INMIC)*, pages 62–68, 2001.

[Joa-Ng and Lu, 1999] M. Joa-Ng and I-Tai Lu. A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks. *IEEE on Selected Areas in Communications*, 17(8):1415–1425, 1999.

[Johnson and Maltz, 1996] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing, edited by Tomasz Imielinski and Hank Korth, Chapter 5*, 1996.

[Karlof and Wagner, 2003] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks*, 1, 2003.

[Karlof *et al.*, 2004] C. Karlof, N. Sastry, and D. Wagner. TinySec: A link layer security architecture for wireless sensor networks. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[Karp and Kung, 2000] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless sensor networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[Kim and Tsudik, 2005] J. Kim and G. Tsudik. Securing route discovery in DSR. In *Proceedings of the IEEE Conference on Mobile and Ubiquitous Systems (Mobiquitous)*, pages 247–258, July 2005.

[Kim *et al.*, 2005] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic routing made practical. In *Proceedings of the ACM Symposium on Networked Systems Design and Implementation (NSDI)*, pages 217–230, 2005.

[Kim *et al.*, 2006] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker. Lazy cross-link removal for geographic routing. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[Kuhn *et al.*, 2003] F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 267–278, 2003.

[Levis *et al.*, 2003] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *In Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSyS)*, 2003.

[Li *et al.*, 2001] Q. Li, J. Aslam, and D. Rus. Hierarchical power-aware routing in sensor networks. In *Proceedings of the DIMACS Workshop on Pervasive Networking*, 2001.

[Liu and Kaiser, 2003] C. Liu and J. Kaiser. A survey of mobile ad hoc network routing protocols. Technical Report 2003-08, Department of Computer Structures, University of Ulm, 2003.

[Manjeshwar and Agarwal, 2001] A. Manjeshwar and D. P. Agarwal. TEEN: a routing protocol for enhanced efficiency in wireless sensor networks. In *Proceedings of the International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, 2001.

[Manjeshwar and Agarwal, 2002] A. Manjeshwar and D. P. Agarwal. APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. In *Proceedings of the Parallel and Distributed Processing Symposium (IPDPS)*, pages 195–202, 2002.

[Mao, 2004] W. Mao. *Modern Cryptography: Theory and Practice*. Prentice Hall PTR, 2004.

[Marshall, 2003] J. Marshall. An analysis of the secure routing protocol for mobile ad hoc network route discovery: using intuitive reasoning and formal verification to identify flaws. msc thesis, department of computer science, florida state university, 2003.

[(MicaZ Datasheet), 2005] CrossBow Technology Inc. (MicaZ Datasheet). Micaz mote platform. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAZ_Datasheet.pdf`, 2005.

[Murthy and Garcia-Luna-Aceves, 1996] S. Murthy and J.J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. In *ACM Mobile Networks and App. J., Special Issue on Routing in Mobile Communication Networks*, pages 183–197, 1996.

[Nikaein *et al.*, 2001] Navid Nikaein, Christian Bonnet, and Neda Nikaein. Harp-hybrid ad hoc routing protocol. In *Proceedings of International Symposium on Telecommunications (IST)*, 2001.

[Osipov, 2007] E. Osipov. tinyLUNAR: One-byte multihop communications through hybrid routing in wireless sensor networks. In *Proceedings of the 7th International Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN)*, 2007.

[Papadimitratos and Haas, 2002] P. Papadimitratos and Z. Haas. Secure routing for ad hoc networks. In *Proceedings of SCS Communication Networks and Distributed Systems Modelling Simulation Conf. (CNDS)*, 2002.

[Perkins and Bhagwat, 1994] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings of the ACM Conference on Communications Architecture, Protocols and Applications (SIGCOMM)*, pages 234–244, 1994.

[Perkins and Royer, 1999] C. Perkins and E. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[Perrig *et al.*, 2000] A. Perrig, R. Canetti, D. Tygar, and D. Song. Efficient authentication and signature of multicast streams over lossy channels. In *Proceedings of the IEEE Symposium on Research in Security and Privacy (Oakland)*, 2000.

[Perrig *et al.*, 2002] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. *Wireless Networks Journal (WINE)*, 8, 2002.

[Pfitzman and Waidner, 2001] B. Pfitzman and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proceedings of the 22nd IEEE Symposium on Security & Privacy*, 2001.

[Piotrowski *et al.*, 2006] K. Piotrowski, P. Langendoerfer, and S. Peter. How public key cryptography influences wireless sensor node lifetime. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, 2006.

[Rivest *et al.*, 1978] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *CACM 21,2*, 1978.

[Sanzgiri *et al.*, 2002] K. Sanzgiri, B. Dahill, B. Levine, C. Shields, and E. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of the International Conference on Network Protocols (ICNP)*, 2002.

[Shah and Rabaey, 2002] R. C. Shah and J. Rabaey. Energy aware routing for low energy ad hoc sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2002.

[Shnayder *et al.*, 2004] Victor Shnayder, Mark Hempstead, Bor rong Chen, Geoff Werner-Allen, , and Matt Welsh. Simulating the power consumption of large-scale sensor network applications. In *In Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, MD, November 2004.

[Shoup, 1999] V. Shoup. On formal models for secure key exchange (version 4). Technical report, revision of IBM Research Report RZ 3120, 1999.

[Singha *et al.*, 1998] S. Singha, M. Woo, and C. Raghavendra. Power-aware routing in mobile ad hoc networks. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 1998.

[TinyOS 2.x, 2007] TinyOS 2.x. `http://www.tinyos.net/tinyos-2.x/doc/`, 2007.

[TinyOS Alliance, 2007] TinyOS Alliance. Powertossim for tinyos 2.x. `http://tinyos.cvs.sourceforge.net/tinyos/tinyos-2.x-contrib/cedt/`, 2007.

[(Tmote Sky Datasheet), 2006] Moteiv Corporation (Tmote Sky Datasheet). Tmote sky – ultra low power ieee 802.15.4 compliant wireless sensor module. `http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf`, 2006.

[Toussaint, 1980] G. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268, 1980.

[Wood *et al.*, 2006] A. D. Wood, L. Fang, J. A. Stankovic, and T. He. SIGF: A family of configurable, secure routing protocols for wireless sensor networks. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, 2006.

[X9.63, 1999] ANSI X9.63. The elliptic curve digital signature algorithm (ECDSA), 1999.

[Xu *et al.*, 2005] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2005.

[Yang and Baras, 2003] S. Yang and J. Baras. Modeling vulnerabilities of ad hoc routing protocols. In *Proceedings of the ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003.

[Yang *et al.*, 2006] H. Yang, S. Wong, S. Lu, and L. Zhang. Secure diffusion for wireless sensor networks. In *IEEE International Conference on Broadband Communications, Networks, and Systems (Broadnets)*, 2006.

[Yu *et al.*, 2001] Y. Yu, D. Estrin, and R. Govindan. Geographical and energy-aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical Report TR-01-0023, University of California, Los Angeles, Computer Science Department, 2001.

[Zapata and Asokan, 2002] M. G. Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, 2002.

[Zhu *et al.*, 2003] S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, 2003.

[Çamtepe and Yener, 2005] S. A. Çamtepe and B. Yener. Key distribution mechanisms for wireless sensor networks: a survey. Technical Report TR-05-07, Rensselaer Polytechnic Institute, Computer Science Department, 2005.