

Executive summary

Routing is a fundamental functionality, therefore hostile interventions aiming to disrupt and degrade the routing service have a serious impact on the overall operation of the entire network. Still, security of routing protocols has fallen beyond the scope of research so far. In the UbiSec&Sens Project, our objective is to overcome this problem by studying routing security and developing a secure routing protocol for wireless sensor networks. This document contains our results related to this objective.

This document is organized into three parts. The first part is concerned with problem statement of secure routing and state-of-the-art. In particular, we informally describe the adversary model, the most general attacks against routing protocols in WSNs, and the countermeasures proposed against some of these attacks. The second part contains the description of our formal security framework for sensor network routing protocols (which serves as a formal specification of the security objectives). In addition, we demonstrate the usage of the model by two illustrative examples: first, we formalize a simple attack against the secured TinyOS beaconing in our model; second, we show that INSENS, which is a link-state routing protocol for wireless sensor network, is a provably secure routing protocol in our model.

Our formal security model for routing protocols in wireless sensor networks is based on the well-known simulation paradigm, but it differs from previously proposed models in several important aspects. First of all, the adversary model is carefully adopted to the specific characteristics of wireless sensor networks. In our model, the adversary is not all-powerful, but it can only interfere with communications within its own radio range. A second important contribution is that we defined the output of the dynamic models that represent the ideal and the real operations of the system as a suitable function of the routing state of the honest nodes, instead of just using the routing state itself as the output. This allows us to model different types of routing protocols in a common framework. In addition, this approach hides the unavoidable distortions caused by the adversary in the routing state, and in this way, it makes our definition of routing security satisfiable.

In the third part, we describe the specification of a particular secure label switching routing protocol for sensor networks, together with its security proof in the proposed formal security framework. This protocol is the secure variant of TinyLUNAR, which is an extensively used routing protocol in UbiSec&Sens. This part also includes the performance analysis of the proposed label switching routing protocol. The analysis was carried out by means of simulations, and this part also describes the simulation environment, the figures of merit, the simulation parameters, and the simulation results. We also compare the performance of this provably secure routing protocol to its non-secure counterpart.

List of authors

Company	Author
BUTE	Levente Buttyán
BUTE	Gergely Ács

Contents

Executive summary	3
List of authors	4
1 Introduction	8
I Attacks and countermeasures	9
2 Attacks on Sensor Network Routing	9
2.1 Adversary model	9
2.2 Objectives of attacks	10
2.3 Attack methods	10
2.4 Specific examples	12
2.4.1 TinyOS beaconing	12
2.4.2 Directed Diffusion	12
2.4.3 GPSR	13
3 Countermeasures	14
3.1 Link layer security (prevention of outsider attacks)	15
3.2 Secure neighbor discovery	15
3.2.1 Preventing the Sybil attack	15
3.2.2 Detecting node replication attacks	17
3.2.3 Detecting wormholes	18
3.3 Authenticated broadcast (prevention of malicious flooding)	21
3.3.1 Digital signatures	21
3.3.2 One-way hash chains	22
3.3.3 μ Tesla	22
3.3.4 One-time signatures	23
3.4 Multi-path routing (to achieve robustness)	24
4 Summary	24
II A formal framework for provably secure routing in WSNs	26
5 The model	26
5.1 Adversary model	26
5.2 Network model	27
5.3 Security objective function	27
5.4 Dynamic model	28
5.5 Definition of secure routing	30
6 Example 1: Insecurity of TinyOS routing	31
6.1 Operation of an authenticated routing protocol	31
6.2 Formalization of a simple attack	32
7 Example 2: Security of INSENS	33
7.1 Operation of INSENS	33
7.2 Security analysis	34
7.3 Discussion	36
8 Summary	36

III Secure-TinyLUNAR: A provably secure routing protocol for WSNs	38
9 Insecurity of TinyLUNAR	38
9.1 Operation of TinyLUNAR	38
9.2 Attacks against TinyLUNAR	40
10 Secure-TinyLUNAR	40
10.1 Operation of Secure-TinyLUNAR	40
10.1.1 Computation and communication overhead	42
10.2 Security analysis	43
10.2.1 The model	43
10.2.2 Tolerable imperfections of the model	44
10.2.3 Proof of Security	45
10.3 Implementation and performance evaluation	46
10.3.1 Module description	46
10.3.2 Simulation environment	47
10.3.3 Figures of merit	48
10.3.4 Simulation results	48
11 Summary	50
12 Related work	51

List of Figures

1	Message modification attack	9
2	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. Initially, the base station B floods the network with a beacon. Receiving this beacon from node E, the adversarial node A rebroadcasts it in the name of node D that is denoted by a dashed arrow. Upon the reception of this falsified beacon, node C will believe that the sender of this beacon is node D. Thus, C sets D as its parent node.	13
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. Node A can allure the traffic by disseminating faked interests in the name of B and/or by manipulating the reinforcement strategy of Directed Diffusion. As a result, all honest nodes that receive the falsified interests originating from A will select the path that traverse node A. This is illustrated in the figure by that all nodes in a considerably part of the network set gradients towards node A. After alluring the traffic, node A can mount selective forwarding.	14
4	Route diversion attack against GPSR. The adversarial nodes are denoted by A and A'. Using GPSR, the source node S sends a packet towards B. Receiving this packet, A alters the destination of the packet from B to A'. When A' receives the packet from K, it recovers the original destination to B. Thus, the packet reaches B along nodes E, A, H, K, A', Q that is denoted by solid arrows. In contrast to this, if the adversary does not divert the packet, it will traverse nodes E, A, C, denoted by dashed arrows, which is a much shorter route.	15
5	Line selected multicast to defend against node replication attack. The single replica of node A is denoted by a black filled node A. Let us assume that the replica passes the plausibility check of its neighbors. Thus, some of these neighbors B, D, E, F choose random geographic locations denoted by ref_B , ref_D , ref_E , ref_F , respectively, and route the location claim of the replica to the nodes (witnesses) closest to these selected locations (routes are denoted by dashed arrows). The location claim is stored at each intermediate node. The neighbors of honest node A, which are J, G, H, perform the same steps (these routes are denoted by dotted arrows). In this example, the replication is detected at node B which receives the location claims of both honest node A and its replica.	18
6	The real-world model.	30
7	A simple attack against ABEM	32
8	Impersonation attacks against TinyLUNAR. Dashed lines denote the neighborhood relations, whereas arrows denote the routing entries.	41
9	Structure of the (Secure) TinyLUNARC component.	47
10	Energy consumption and network delay measurements in case of symmetric links. The whiskers on the top of the bars correspond to the 95% confidence interval of the average. . . .	50
10	Energy consumption and network delay measurements in case of symmetric links. The whiskers on the top of the bars correspond to the 95% confidence interval of the average. . . .	51
11	Energy consumption and network delay measurements in case of asymmetric links. The whiskers on the top of the bars correspond to the 95% confidence interval of the average. . . .	52
11	Energy consumption and network delay measurements in case of asymmetric links. The whiskers on the top of the bars correspond to the 95% confidence interval of the average. . . .	53

1 Introduction

Routing is a fundamental function in every network that is based on multi-hop communications, and wireless sensor networks are no exceptions. Consequently, a multitude of routing protocols have been proposed for sensor networks in the recent past. However, most of these protocols have not been designed with security requirements in mind. This means that they can badly fail in hostile environments. Paradoxically, research on wireless sensor networks have been mainly fueled 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.

We 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, exemplified in [38], is to list different types of possible attacks against routing in wireless sensor 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 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 has 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 [38], which are potentially useful 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, we propose to base the design of secure routing protocols for wireless sensor networks on a formal security model. More precisely, in this document, we introduce a formal framework which allows us to define what we really mean by security of routing in the context of wireless sensor networks, and to prove if a given protocol satisfies the definition. We illustrate the usefulness of this formal framework by proposing a secure label switching routing protocol for wireless sensor networks and formally proving that it is indeed secure in our model. In addition, we study the performance of the proposed protocol by means of simulations, and we report on a prototype implementation of the protocol for MicaZ and Mica2 sensor nodes.

The rest of this document is organized into three parts. The first part is concerned with problem statement of secure routing and state-of-the-art. In particular, we informally describe the adversary model, the most general attacks against routing protocols in WSNs, and the countermeasures proposed against some of these attacks. The second part contains the description of our formal security framework for sensor network routing protocols (which serves as a formal specification of the security objectives). In addition, we demonstrate the usage of the model by two illustrative examples: first, we formalize a simple attack against TinyOS beaconing in our model; second, we show that INSENS, which is link-state routing protocol for wireless sensor network, is a provably secure routing protocol in our model. Finally, in the third part, we describe the specification of a particular secure label switching routing protocol for sensor networks, together with its security proof in the proposed formal security framework. This protocol is the secure variant of TinyLUNAR, which is an extensively used routing protocol in UbiSec&Sens. This part also includes the performance analysis of the proposed label switching routing protocol. The analysis was carried out by means of simulations, and this part also describes the simulation environment, the figures of merit, the simulation parameters, and the simulation results. We also compare the performance of this provably secure routing protocol to its non-secure counterpart.

The results presented in this document were published in the following papers: [94] [92] [96] [97] [98] [93].

Part I

Attacks and countermeasures

2 Attacks on Sensor Network Routing

2.1 Adversary model

The adversary can mount her attacks from 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, she 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. The sensor-class devices and laptop-class devices that are under the control of the adversary are further called adversarial nodes. 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.

All attacks performed by the adversary consist of simple message manipulations like message injection, deletion, modification, re-ordering, and simply relaying messages without following the routing protocol rules faithfully. Before investigating more sophisticated attacks that employ the previously listed message manipulations, we 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 [85]. 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 prevents the receiver from receiving messages, while at the same time allows 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 the adversarial node. Here, we sketch two scenarios for message modification, which are illustrated on Figure 1. By these simple examples, we intend to point out the feasibility of message modification assuming even direct communication between the sender and the receiver node. We assume that communication range implies interference range, and vice-versa.

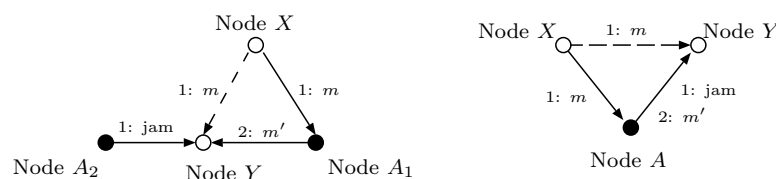


Figure 1: Message modification performed by the cooperation of two adversarial nodes A_1 and A_2 (on the right-hand side) in Scenario 1, and employing overhearing, jamming, and relaying with a single adversarial node A (on the left-hand side) in Scenario 2. Honest nodes are labelled by X and Y . Arrows 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.

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 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 [26]). The adversary can exploit this CRC mechanism to modify a message in the following way (illustrated on Figure 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.

The feasibility of jamming attacks is studied and demonstrated in [85]. Although, the authors conclude in that paper that the success of jamming attacks mainly depend on the distance of the honest nodes and the jammer node, various jamming techniques has been presented there that can severely interfere with the normal operation of the network.

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

2.2 Objectives of attacks

Generally speaking, the adversary primarily intends to thwart the objectives of routing protocols. More specifically, she wants to degrade the performance of routing, or ultimately, she may attempt to completely disrupt the routing service and cause network malfunctioning. 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 she can easily degrade the packet delivery ratio or increase the network delay by dropping packets and delaying packet forwarding).

2.3 Attack methods

In the previous subsections, we described the capabilities of the adversary in sensor networks and the general objectives of the attacks launched by this adversary. Now, we give an overview of the specific attack methods that can be used by the adversary in order to achieve her objectives.

The simplest attack methods are composed of the basic message manipulations techniques. This includes *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 consume valuable network resources at honest sensor nodes (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 have trivial effects: in this way, the adversary can separate some of the nodes from the base station that can only reach the base station 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. We 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.

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 is a protocol-dependant attack. 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 will be described in Subsection 2.4) 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 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 she 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.

Finally, there are other attacks that are mainly related to neighbor discovery, such as the *wormhole attack*, the *Sybil attack*, and the *node replication attack*. We consider these attacks against routing, since many sensor network 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 denial-of-service 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 [22], a single adversarial node illegitimately uses *multiple* identities during the routing process. This can have devastating effects on multipath routing protocols [38], 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 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 she may be able to attract all traffic to her; this is often referred to as the *sinkhole* attack [38].

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 [38]. Some of these will be exemplified in the following subsection.

2.4 Specific examples

In this subsection, we illustrate the previously described attack methods on different types of sensor network routing protocols: we show how the adversary can subvert TinyOS beaconing, Directed Diffusion and GPSR.

2.4.1 TinyOS beaconing

Originally, the authors of TinyOS proposed a very simple routing protocol in [30], 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.

In the following, we show how the adversary described in Subsection 2.1 can create a routing loop in a sensor network using TinyOS beaconing. Let us consider Figure 2. 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.

2.4.2 Directed Diffusion

Directed Diffusion [35] is another mainstream topology-based routing protocol. 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.

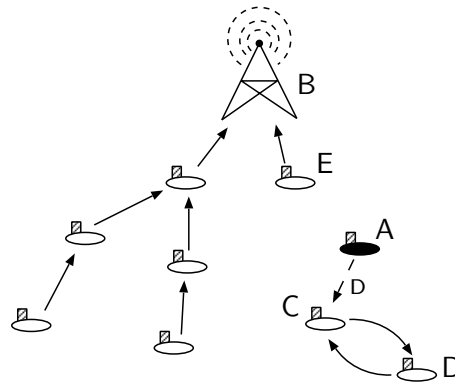


Figure 2: Creating a routing loop in TinyOS beaconing. The only adversarial node is denoted by A. For each node, the solid line points to the parent node. Initially, the base station B floods the network with a beacon. Receiving this beacon from node E, the adversarial node A rebroadcasts it in the name of node D that is denoted by a dashed arrow. Upon the reception of this falsified beacon, node C will believe that the sender of this beacon is node D. Thus, C sets D as its parent node.

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 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 [35].

The adversary can easily mount black- and grayhole attacks against Directed Diffusion. Blackhole attack means that the adversary allures all traffic from a particular area along an adversarial node, and then drops all received packets. Grayhole attack is 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 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.

2.4.3 GPSR

GPSR (Greedy Perimeter Stateless Routing) [39] 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 sensor 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 GG (Gabriel Graph) [27], RNG (Relative Neighborhood Graph) [77], CLDP (Crossing Link Detection Protocol) [40] or LCLR (Lazy Cross Link Removal) [41]. 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 l checks

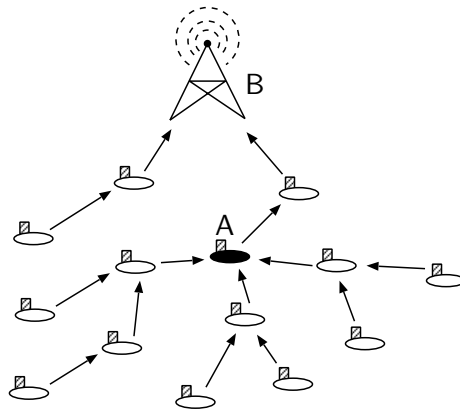


Figure 3: Black- and grayhole attack against Directed Diffusion. The only adversarial node is denoted by A. For each node, the solid lines denote the gradients set towards the base station. Node A can allure the traffic by disseminating faked interests in the name of B and/or by manipulating the reinforcement strategy of Directed Diffusion. As a result, all honest nodes that receive the falsified interests originating from A will select the path that traverse node A. This is illustrated in the figure by that all nodes in a considerably part of the network set gradients towards node A. After alluring the traffic, node A can mount selective forwarding.

whether it has a neighbor that is closer to node D than itself. If it has, the message is forwarded to that node. Otherwise, l 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 l and the destination, denoted by d . After putting its own location into the packet, l 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 l . In the latter case, the node that is closer to D than l performs the same steps that l 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 l is traversed again) without reaching a node being closer to D, then the packet is marked as undeliverable.

We 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 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 E, A, H, K, A' , Q instead of nodes E, A, C that would be a much shorter and less energy consuming route.

3 Countermeasures

In this section, we review some countermeasures that can be used to defend against the attacks described in Subsection 2.3. Some of these countermeasures will serve as building blocks for secure sensor network routing protocols, which will be described in the next section. In Subsection 3.1, we address link layer security focusing on how one can ensure message authenticity, confidentiality and freshness at the link layer. In Subsection 3.2, we consider the security of neighbor discovery; we describe the basic defenses against the Sybil, the node replication and the wormhole attacks. The techniques of broadcast authentication are discussed in Subsection 3.3. Finally, we are concerned with robust data delivery in Subsection 3.4.

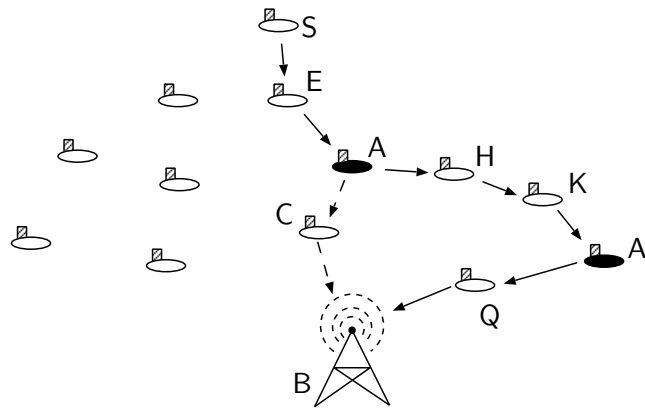


Figure 4: Route diversion attack against GPSR. The adversarial nodes are denoted by A and A'. Using GPSR, the source node S sends a packet towards B. Receiving this packet, A alters the destination of the packet from B to A'. When A' receives the packet from K, it recovers the original destination to B. Thus, the packet reaches B along nodes E, A, H, K, A', Q that is denoted by solid arrows. In contrast to this, if the adversary does not divert the packet, it will traverse nodes E, A, C, denoted by dashed arrows, which is a much shorter route.

3.1 Link layer security (prevention of outsider attacks)

To protect against malicious manipulations of routing messages exchanged between sensor nodes as well as between base station and sensor nodes, one can employ traditional cryptographic primitives, whereby one can provide authenticity in conjunction with integrity, as well as confidentiality for messages. However, providing these security services in an end-to-end manner is not desirable in sensor networks, because the authenticity of routing control messages need to be verified and their content need to be accessed by intermediate nodes. Therefore, we need to provide these services in a hop-by-hop manner at the link layer.

Providing link layer security is quite straightforward, however, one must keep in mind the resource constraints of the sensor nodes. In particular, sensor nodes are less capable to perform expensive cryptographic computations and communicate long signatures and MACs. For this reason, symmetric key cryptographic mechanisms are often preferable in sensor networks.

An example of a widely used software package that provides symmetric key cryptographic mechanisms and that can be used to implement security services at the link layer is TinySec [37], which runs on top of TinyOS [30].

Finally, we note that the countermeasures implemented at the link layer do not defend against insider attacks, where the adversary controlling some intermediate nodes can manipulate messages in an undetectable way.

3.2 Secure neighbor discovery

3.2.1 Preventing the Sybil attack

A Sybil node, which uses multiple identities, can obtain these identities by either fabricating new identities or stealing existing ones from honest nodes (e.g., by node capturing). Current defense mechanisms can be grouped into two major categories; *decentralized* and *centralized* techniques. The former includes resource testing, cryptographic defenses, and secure position verification, while the latter incorporates registration and anomaly detection.

Resource testing relies on the observation [22] that each node (device) is generally limited in some resources (e.g., computation, storage, or communication). Even an adversarial node that uses multiple fake identities cannot multiply its resources, therefore, with appropriate mechanisms one can discover that those fake identities do not belong to different nodes. For instance, one feasible way to do this is the radio resource testing [57], which assumes that every node has only a single antenna that is generally incapable of using multiple channels simultaneously. Applying this scheme, a verifier node assigns different channels to all its neighbors. Then, the verifier selects randomly a channel and broadcasts a message on that. If the neighbor assigned to this channel

is legitimate, it should answer or at least acknowledges the message. However, if the channel is assigned to a Sybil node, the corresponding adversarial node will not respond with a certain probability, because it has only one antenna on which he can listen on a single channel. Repeating this test, the probability that the misdeed is not detected can be made arbitrarily small [57], even if the verifier can test only a subset of its neighbors at one time (i.e., the number of channels that can be used is less than the number of the neighbors).

One advantage of all resource testing schemes is that they can be used against both fabricated and stolen identities. By contrast, all cryptographic defenses can only prevent the illegal fabrication of identities. However, if the adversary retrieves somehow the cryptographic material (e.g., secret keys) needed to authenticate an existing identity (i.e., she steals the identity), then this identity can be used by a Sybil node.

Some cryptographic defenses are based on the *random key predistribution* scheme proposed in [25]. In this approach, each node is randomly assigned a subset of symmetric keys from a common key pool. If the size of this subset is large enough, any two nodes will have at least one common shared key with high probability based on the birthday paradox. The adversary who aims at fabricating a *new* identity can easily do that by capturing multiple nodes and use every combination of the compromised keys. In [57], the authors propose a defense by binding keys to identities. Here, we only outline the main idea: All keys are indexed in the common key pool with m keys, and the identity ID is assigned to k keys by calculating the key index with $F_{h(ID)}^{PRF}(i)$ ($1 \leq i \leq k$), where h is an one-way hash function and F^{PRF} is a Pseudo-Random Function (e.g., F^{PRF} can be a CBC-MAC construction [62]). Therefore, the best that an adversary can do is to perform an exhaustive search on the set of fabricated identifiers (i.e., calculating $F_{h(ID')}^{PRF}(i)$ to find a valid ID') until she finds a candidate such that all keys assigned to the candidate are known by the adversary. This scheme provides a reasonable level of security until the number of captured nodes reaches a certain threshold. However, above this threshold, the adversary can easily fabricate new identities despite using this scheme. Single space pairwise key distribution [7, 6] approaches are inherently resistant against identity fabrication until no more than λ nodes are compromised due to the λ -secure property. If more than λ nodes are compromised, the adversary will be able to fabricate arbitrary number of identities. Single space approaches and the basic pool approach are combined in the multi-space pairwise key distribution approaches [23, 48]. Here, k out of m key spaces are randomly assigned to each node, and the adversary must compromise more than λ instances of each space to compromise that only one key space. In [57], the authors showed that multi-space pairwise approaches provide stronger defense against identity fabrication than both basic key pool and single space approaches (i.e., by multi-space scheme, the adversary needs to compromise more nodes to successfully fabricate a new identity than in the basic key pool or single space schemes).

Further cryptographic defenses can be provided by using some *identity certification method* like digital certificates. However, these schemes are based on digital signatures that are considered to be expensive in terms of computation and communication. A more viable identity certificate method was proposed in [89] that uses Merkle-trees [54] and one-way hash chains instead of expensive public key cryptography to prove identities.

There are also some additional decentralized defenses like *secure position verification* [69] and *code attestation* (a.k.a, remote code verification) [58, 76]. The former has shown significant progress recently, but the latter one still incurs substantial overhead.

In case of the centralized approaches, such as registration and anomaly detection, a trusted entity (e.g., the base station) is required to have a global view of the network. In case of the *registration* approach, this central entity can preclude Sybil attacks by accurately tracking the operation of the network (e.g., a node can verify its neighbors by querying the central entity for the list of nodes registered in the network). Alternatively, this central entity can also detect *anomaly* related to Sybil attacks (e.g., nodes which are reported to have too many neighbors are good candidates for Sybil nodes). All these centralized schemes may provide a reasonable solution against Sybil attacks in many simple scenarios (e.g., in small-sized networks).

Summarizing all countermeasures against Sybil attacks, we can conclude that none of them provide a perfect defense. The radio resource verification may be circumvented by using special radio hardware and it can also incur substantial energy consumption. All centralized techniques has limited applicability to sensor networks, since the central element becomes a single point of failure and these solutions are not scalable in general. Moreover, cryptographic solutions cannot be used against stolen identities and the success of secure position verification highly depends on the accuracy of the localization method.

3.2.2 Detecting node replication attacks

Node replication occurs when a single identity is used by multiple nodes simultaneously in the network. For instance, the adversary may try to deploy additional adversarial nodes that use the same identity or an identity that is already in use by a honest node in the network. Note that Sybil attacks and node replication attacks can be jointly mounted at the same time; a single adversarial node can use multiple identities from which at least one is already in use simultaneously by further adversarial nodes or honest nodes.

There exist centralized and decentralized detection algorithms for node replication attacks. By centralized detection, each node sends its whole neighborlist to the base station which then can filter out replicated nodes and revoke them by flooding the network with authenticated revocation lists [25]. However, similar to the centralized methods in the case of Sybil attacks, this method also has many drawbacks such as scalability problems and degraded robustness. Decentralized detection protocols are much promising due to the distributive nature of sensor networks. Here, we briefly present two decentralized approaches with illustration purposes: randomized multicast [10] and line selected multicast [10].

Both randomized multicast and line selected multicast operates similarly; the location claim of all nodes are disseminated in the network to some witness nodes which record all received location claims. If a witness node receives duplicate location claims with the same originator identity but with different locations, then the replication is detected and the witness floods the whole network with an authenticated revocation message. Here, we assume that each node is aware of its location and also knows the size of the communication range.

Using randomized multicast detection, all nodes in the network broadcast their location claims. The location claim is authenticated by each node using public key cryptography or another more efficient broadcast authentication scheme described in Subsection 3.3. The location claim of node A is received by the neighbors of A. Then, each neighbor of A verifies the authenticity and the plausibility of the location claimed by A (e.g., the plausibility can be checked by calculating the distance between the claimed location and the location of the neighbor, and if this distance is less than the radius of the communication range, then the location claim is plausible, otherwise, it is considered to be implausible). If the location claim is not authentic or it is implausible, the neighbor discards the location claim. Otherwise, with probability p , each neighbor selects r random locations in the network and for each of these locations, the neighbor uses geographic routing to forward the location claim of A to the node that is closest to the selected location (e.g., by using the routing algorithm in GHT [66]). If the average number of a node's neighbors is denoted by k , then $p \cdot r \cdot k$ is a good approximation for the number of nodes that receive A's location claim. If $r \approx \sqrt{n}$ (i.e., $p \cdot r \cdot k$ is in the order of $O(\sqrt{n})$), where n denotes the number of all nodes in the network, then based on the birthday paradox a single replication can be detected with high probability (≈ 0.5) [10] (i.e., there exists at least one witness node that receives the location claims from both the original and the replicated node with high probability). Moreover, as the number of the replication of the same node increases so does the probability of the detection (assuming that each node has at least one legitimate neighbor). On average, each node needs to store $p \cdot r \cdot k$ location claims, thus the storage cost is in the order of $O(\sqrt{n})$, and the communication cost is in the order of $O(n\sqrt{n} \cdot p \cdot r \cdot k) = O(n^2)$ [10]. However, the storage cost can be further reduced by using some time synchronization enhancements detailed in [10].

In order to reduce the communication cost of the randomized multicast detection, another scheme called line selected multicast was proposed in [10] that was inspired by Rumor Routing [8]. The operation of line selected multicast is illustrated in Figure 5. The idea is that along the path between a neighbor of A and a witness node each intermediate node verifies whether it has already received a different location claim with the identity of A. If the intermediate node has multiple location claims with A's identity, then it floods the network with an authenticated revocation message. Otherwise, the node buffers the location claim (if there is no any location claim with identity A in the buffer), and forwards the location claim towards the witness node. In this way, there will be line segments throughout the network that radiate out in random directions from node A. Now, if there is a replicated node with identity A in the network, the probability that the replication is detected equals to the probability that at least two line segments radiating out from A and its replica, respectively, intersect. It can be shown by using Monte-Carlo simulations that the probability that two random line segments originating from A and its replica intersect is greater than 0.56. If there are five line segments per node this probability can further increase even up to 0.95 depending on the network topology. If each line segments has a length in the order of $O(\sqrt{n})$, the communication cost of this scheme is $O(n\sqrt{n})$ for the whole network. Regarding the storage cost, each node stores $O(\sqrt{n})$ location claims. Moreover, the storage cost can be further reduced by

using some time synchronization enhancements detailed in [10].

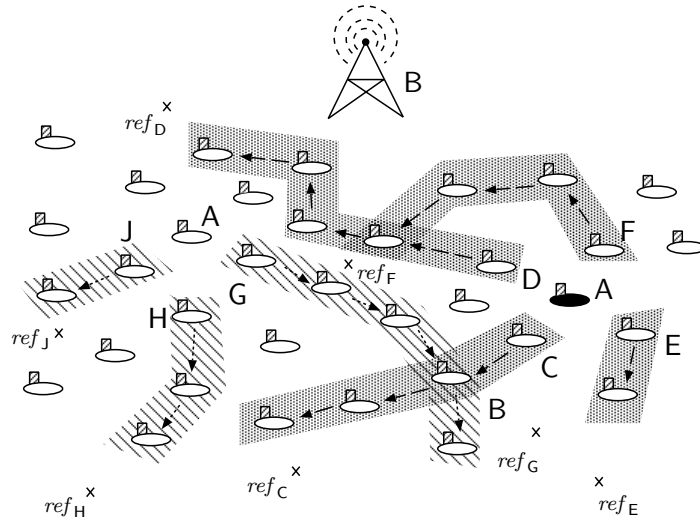


Figure 5: Line selected multicast to defend against node replication attack. The single replica of node A is denoted by a black filled node A. Let us assume that the replica passes the plausibility check of its neighbors. Thus, some of these neighbors B, D, E, F choose random geographic locations denoted by ref_B , ref_D , ref_E , ref_F , respectively, and route the location claim of the replica to the nodes (witnesses) closest to these selected locations (routes are denoted by dashed arrows). The location claim is stored at each intermediate node. The neighbors of honest node A, which are J, G, H, perform the same steps (these routes are denoted by dotted arrows). In this example, the replication is detected at node B which receives the location claims of both honest node A and its replica.

Both schemes rely on the assumption that node A has at least one honest neighbor that faithfully forwards the location claim of A. However, this assumption does not hold for every practical scenario (e.g., the adversary may compromise all neighbors of A). To circumvent this problem, the authors in [10] proposed a slightly modified scheme where not the immediate neighbors of A but the closest honest nodes to A will act as pseudo-neighbors of A (i.e., they explicitly query A for its location claim, and in case they do not receive any response, they deny to forward any traffic originating from A). More interested readers are referred to [10].

We note that randomized multicast detection as well as line selected multicast detection rely on some existing routing infrastructure like some variant of GPSR. Thus, the effectiveness of the schemes depends on two factors: (i) how fast the misdeed is detected by the sensor nodes, (ii) how fast the revocation list is distributed *before* routing any data packets. Finally, we note that these schemes are not applicable to those protocols that use locally unique identities, because in that case, two nodes being far away from each other can legally use the same identity.

3.2.3 Detecting wormholes

Broadly speaking, the different wormhole detection mechanisms fall into two classes: the centralized mechanisms and the decentralized ones. In case of the centralized approach, data collected from the local neighborhood of every node are sent to the base station. The base station uses the received data to construct a model of the entire network, and tries to detect inconsistencies in this model that are potential indicators of wormholes. In the decentralized approach, each node constructs a model of its own neighborhood using locally collected data, and performs wormhole detection locally. The advantage of the decentralized wormhole detection mechanisms is that they are more energy efficient and scale better than the centralized mechanisms, therefore, they can be used in a wider range of applications. Their disadvantage is that they usually rely on some strong system assumptions (e.g., accurate time synchronization between the nodes) or on the availability of some special hardware in the nodes (e.g., a GPS receiver or a directional antenna).

Two examples of the centralized approach are the wormhole detection mechanisms proposed in [12] and [82]. In the scheme described in [12], the nodes report only the list of their believed neighbors to the base sta-

tion, and the model constructed by the base station consists of the connectivity graph of the network. A crucial observation is that a wormhole *always* increases the number of edges in the connectivity graph, as it introduces new neighbor relationships. This increase in the number of edges changes the properties of the connectivity graph in a detectable way with respect to some expectations that are based on some basic assumptions about the system (e.g., the distribution of node positions, the communication range of the nodes, etc). The main idea is to detect the changes in the connectivity graph using statistical hypothesis testing methods. In particular, wormholes are detected in [12] by identifying distortions in the distribution of the number of neighbors and in the distribution of the length of the shortest paths between all pairs of nodes using the χ^2 -test.

In [82], the nodes also estimate the distances from their believed neighbors, and send their neighbor list with the estimated distances to the base station. In this case, the model constructed by the base station is a virtual layout of the network. The crucial observation here is that a wormhole contracts the virtual layout in certain regions, since it makes some nodes appearing neighbors while in reality these nodes are far away from each other. The main idea is then to detect these contractions by visualizing the virtual layout.

Examples of the decentralized wormhole detection approach can be found in [34]. Most of these decentralized approaches are based on the straightforward idea of estimating the real physical distance between the nodes that are believed to be neighbors. If the estimated distance is larger than the nodes' communication range, then the nodes are likely connected through a wormhole.

Two mechanisms for distance estimation with the specific purpose of wormhole detection are proposed in [34]; they are called geographical and temporal *packet leashes*, respectively. The main idea of both mechanisms is to add some extra information to the packets that restricts their maximum allowed transmission distance.

A geographical leash is based on location information, and it allows the receiver of the packet to determine an upper bound on its distance to the sender. It is assumed that each node is aware of its own location, which may be determined using GPS or some other positioning mechanism. It is further assumed that the nodes maintain loosely synchronized clocks.

A temporal leash is based on timing information, and it ensures that the packet has an upper bound on its lifetime. Indirectly, however, this also ensures an upper bound on the distance between the sender and the receiver, since the packet cannot travel faster than the speed of light. Temporal leashes require that the nodes have tightly synchronized clocks, such that the maximum difference between any two nodes' clocks is in the order of a few microseconds or even hundreds of nanoseconds.

Both geographical and temporal leashes require that the packets carrying the leashes are authenticated and their integrity is protected, since otherwise an adversary can modify a leash and jeopardize the distance estimation. Origin authentication and integrity protection can be based on digital signatures or on symmetric key MACs. The advantage of digital signatures is that they provide broadcast authentication, and therefore, they can be used efficiently for protecting neighbor discovery beacons and route discovery messages that are usually broadcast messages. The disadvantage of digital signatures is that they are several orders of magnitude slower than symmetric key MAC computations, and speed is critical, especially in the case of temporal leashes. In order to overcome this problem, in [34], the authors propose to use μ TESLA with Instant Key-disclosure (TIK) to authenticate temporal leashes in packets.

Both types of leashes can be used for wormhole detection, because they allow the receiver of the packet to detect if the sender is further away than the nodes' communication range. More precisely, the receiver can determine only an upper bound on its distance to the sender. However, if this upper bound is greater than the nodes' communication range, then the receiver should not accept the packet. In this way, packets that arrive through a wormhole are always rejected.

Packet leashes can be added to the neighbor discovery beacons or to the packets of the neighbor discovery protocol when the nodes use such mechanisms explicitly for setting up their neighbor relationships. In that case, the application of packet leashes prevents the establishment of fake neighbor relationships. When no explicit neighbor discovery mechanism is used in the system, packet leashes can still be added to the packets of the routing protocol, in order to prevent the undesirable effects of wormholes on routing.

Another approach that is also based on distance estimation between the nodes, but does not require any clock synchronization or localization mechanisms, is proposed in [79]. This approach is based on the concept of *distance-bounding*, which was first introduced by Brands and Chaum in [9]. Distance bounding is based on the facts that electro-magnetic waves propagate nearly with the speed of light and with current technology it is easy to measure local timings with nanosecond precision. Brands and Chaum's technique essentially consists

of a series of rapid bit exchanges between the two nodes. Each bit sent by the first node is considered to be a challenge for which the other node is required to send a one bit response immediately. By locally measuring the time between sending out the challenges and receiving the responses, the first node can estimate its distance to the other node, assuming that the messages travel with the speed of light and the processing delay at the other node is zero.

Note that the estimated distance is only an upper bound on the real distance between the nodes, because the second node may be closer, but it may delay the responses in order to appear to be farther. Even if the nodes are trusted for not delaying their responses, an active adversary can delay the messages between the parties, and hence, the estimated distance will still be just an upper bound on the real distance. However, in case of a wormhole attack, the adversary's goal is not to make the two nodes believe that they are far away from each other, but on the contrary, the adversary wants that the two nodes believe that they are within each other's range, while in reality they are not. But in order to achieve that the estimated distance is smaller than the nodes' real distance, the adversary should arrange that the messages travel faster than the speed of light, which is impossible. Thus, distance-bounding can be used for wormhole detection.

In [79], a slightly modified version of the above described distance-bounding technique is proposed, which is called Mutual Authenticated Distance-bounding, or shortly MAD. As its name suggests, the MAD protocol allows both nodes to measure the distance between them simultaneously. In addition, it uses symmetric key cryptographic primitives for authentication purposes. In order for this to work, it is assumed that each pair of nodes share a symmetric key, which is established before running MAD between them. Extensions of this idea can be found in [80, 81].

Another wormhole detection mechanism that uses location information is described in [45]. However, the advantage of this mechanism compared to geographical packet leashes is that it requires only a few specialized nodes to be aware of their locations. These specialized nodes are called *guards*, and their role is to help other nodes to set up their neighbor relationships in a secure way.

In [45], it is assumed that the guards have a larger transmission range than that of regular sensor nodes. Let us denote the former by R . Two nodes consider each other neighbors only if they hear each other, and in addition, they hear more than a threshold number of common guards. The messages originating from the guards contain the location information of their sources. The nodes can use this location information to detect wormhole attacks based on the following two principles:

1. Since any guard heard by a node must lie within a range of radius R around the node, a node cannot hear two guards that are $2R$ apart from each other.
2. Normally, a node should not receive the same message twice from the same guard.

It is shown in [45] that if there is a wormhole in the system, then at least one of the above principles is violated with probability close to one. This means that the wormhole is detected in a reliable way.

In [31], the authors propose to use directional information of messages to mitigate wormhole attacks. It is assumed that each node in the network is equipped with a directional antenna. Every such antenna has n , non-overlapping zones, and each zone has a spanning angle of $2\pi/n$; hence, the zones collectively cover the entire area around a node. When a node is idle, it listens to the carrier in omni-directional mode. When it receives a message, it determines the zone in which the received signal strength is maximal, and uses that zone to communicate with the sender. An important assumption is that the orientation of the zones is always established with respect to the Earth's median, and therefore, all nodes use the same orientation no matter of their physical locations and their own orientations. This can be achieved in modern antennas with the help of a magnetic needle that always remains collinear to the Earth's magnetic field.

The main idea is that when two nodes are within each other's communication range, they must hear each other's transmission from opposite directions. However, if the nodes communicate through a wormhole, then this condition is not always satisfied. Unfortunately, this simple principle is not sufficient to reliably detect wormholes as the wormhole can be placed in such a way that two nodes hear each other from opposite directions even though they are communicating through the wormhole. For this reason, the authors of [31] introduce the notion of *verifier* nodes that can help to detect and avoid this situation.

There are specific conditions that a node should satisfy to qualify as a verifier, and it is possible that two nodes are within each other's communication range, but there are no potential verifier nodes that they can use. In that case, the nodes cannot set up a neighbor relationship and we lose a potential link. Unfortunately, losing

links is not desirable, because it reduces the robustness of the network in case of link failures, and increases the average length of the routes in the network. Another disadvantage of this approach is that it requires directional antennas in the nodes, which may be expensive.

3.3 Authenticated broadcast (prevention of malicious flooding)

As many routing protocols rely on flooding or broadcasting routing information, authentication of broadcast data sent by the base station (or rarely by sensor nodes) is a fundamental issue. As we have already seen, the adversary can easily inject extra packets and modify existing ones during routing procedures. Thus, the receiver of a packet should be ensured that the packet is indeed originated from the claimed sender (*source authentication*) and the packet is not altered during the transit (*data authentication*).

The first naive solution would be to employ pairwise shared symmetric keys (i.e., the sender shares a pairwise key with each receiver, and it computes a MAC for each receiver using the corresponding pairwise key). However, this solution is impractical in case the number of receivers is large because the sender may have to attach a huge number of MACs in a single message that is not tolerable due to the high communication cost.

Another simple idea might be to use a network-wide symmetric key. However, in that case, the adversary compromising a node could retrieve the global key, and thus, the adversary would be able to forge messages from the sender. However, if some secure group re-keying mechanism is also provided to refresh network-wide keys in each node either in a periodic or an on-demand manner like in LEAP [90], this can also be an appropriate alternative to perform broadcast authentication. A special case of broadcast authentication is when the receivers are limited to the immediate neighbors of the sender. In that case, a local broadcast key [90] shared with all the one-hop neighbors of the sender can be used to authenticate local broadcast messages.

In the following, we review some asymmetric techniques that are employed to authenticate broadcast messages in sensor networks; digital signatures, one-way hash chains, μ Tesla, and one-time signatures.

3.3.1 Digital signatures

The most common way to authenticate broadcast messages is to apply some digital signature schemes like RSA [67] or ECDSA [84]. Employing RSA is reasonably secure with 1024 and 2048 bit keys but the signature size is quite long compared to the typical message size. Moreover, RSA requires extensive computational effort on behalf of the sender (signer). Elliptic Curve Cryptography (ECC) [56, 42] can provide another signature scheme called ECDSA (Elliptic Curve Digital Signature Algorithm) with the same security level as RSA but with significantly lower signature size. For instance, ECC with 160 bits public key offers the same security as the RSA with 1024 bits key, where the ECC signature size is about 41 bytes compared to the 128 bytes long RSA signature. ECDSA requires less computational effort from the sender (signer) but puts more burden to the receiver (verifier).

Recent empirical studies showed that ECDSA signature generation/verification and RSA signature verification can be effectively implemented on various sensor nodes. The clear advantage of using RSA signature scheme is the fast (and less energy consuming, see [64]) verification compared to ECDSA. As the signature verification is mostly performed by the constrained sensor nodes, the energy consumption of this verification process is a critical issue. As long as the RSA signature is generated on the base station, RSA signature computations incur less overhead with respect to sensor nodes but bears higher communication cost due its larger signature size than ECDSA. Moreover, receiving and sending an RSA signature consumes about 3 percent of energy needed to verify an ECDSA-160 signature on average [64]. As a conclusion, there cannot be made an ultimate decision between RSA and ECDSA regarding broadcast authentication. ECDSA may be more favorable in those applications where the signature is generated by sensor nodes, and/or the communication costs of RSA signatures exceed the verification costs of ECDSA signatures.

Although recent advances in public key cryptography (PKC) of sensor networks are very promising, PKC still falls behind the standard symmetric cryptography approaches in terms of computational performance; the signature verification and generation are still much slower than MAC verification and generation, respectively. Hence, researchers have also proposed other alternatives for broadcast authentication based on symmetric cryptography that we will discuss below.

3.3.2 One-way hash chains

Using one-way hash chains [44] is the most straightforward technique to provide source authentication. By employing this scheme, the sender generates a collection of values (c_0, \dots, c_n) such that each value c_i is a one-way function of the next value c_{i+1} . c_n is also referred to as the seed of the chain. For instance, the sender can derive a seed denoted by c_n from a secret key K_{pr} by applying a Pseudo-Random Function F^{PRF} to a constant value using K_{pr} (i.e., $F_{K_{pr}}^{PRF}(0) = c_n$). Afterwards, the sender generates a hash-chain with length n by iteratively applying a public one-way hash function h to c_n n times, where the i th element of the chain equals to $c_i = h^{(n-i)}(c_n)$, for $0 \leq i < n$. Then, c_0 , also called as the commitment of the chain, is stored at each receiver node. When the sender sends broadcast messages, it places c_1 into the first broadcast message, and c_2 into the second broadcast messages, and so on. A node receiving a broadcast message containing hash value w can check whether there exists j ($1 \leq j \leq n$) such that $c_0 = h^{(j)}(w)$, as the receiver has c_0 . If there exists such j , the receiver node is assured that w is generated by the sender, otherwise, the adversary should invert h that is computationally infeasible due to the one-way property of h .

This authentication technique is favorable in sensor networks due to its generally low resource demand; the base station can generate a hash chain, and the sensor nodes can check the authenticity of the base station if they are provided by the seed beforehand. On the other hand, the receiver has to perform $n - j$ computations to verify c_j if c_i is given, which can be expensive for a constrained sensor node if $j - i$ is large. This motivated more efficient hash chain constructions that were proposed in [87, 70, 15]. Moreover, using hash-chains generally does not guarantee data integrity, unless all messages to be sent are known by the sender a priori (i.e., the message contents are involved in the chain construction), which only holds for a few applications.

3.3.3 μ Tesla

μ Tesla is an efficient broadcast authentication protocol [62] that uses symmetric cryptography (MACs) and hash chains to provide authenticity for the sender. μ Tesla uses *time* to provide asymmetry for the sender which requires that all receivers are loosely synchronized.

Initially, the sender splits up the time into time intervals where each time interval has a constant duration denoted by T_{int} , and creates a one-way hash chain, as described in the previous subsection, where each element of the hash chain c_i corresponds to a secret key K_i . We recall that the commitment of the hash chain (i.e., c_0) is stored at every receiver node. Subsequently, the sender assigns K_i to interval I_i and defines a disclosure lag d for keys, which is typically in the order of few time intervals. We assume that along with the seed of the hash chain the key disclosure schedule including d and T_{int} is also stored at every receiver node.

After the initialization, the sender can authenticate broadcast messages in the i th interval in the following way. The sender computes a MAC on the message content using the key K_i , and appends the MAC to the message. Additionally, the sender may attach a key to the message that can be disclosed (i.e., in interval I_i key K_{i-d} can be disclosed). Of course, the key disclosure lag d highly depends on the maximum synchronization error denoted by δ . In particular, it must be assured that after a key K_i is disclosed by the sender, none of the receivers accept any messages authenticated by key K_i . Hence, each receiver that receives a broadcast message must verify the following security condition: $\lfloor \frac{T_{current} + \delta - T_0}{T_{int}} \rfloor < I_i + d$, where $T_{current}$ is the local time at the verifier node and T_0 is the start time of the first interval. In other words, each receiver checks whether the key used to generate the MAC carried by the message is still secret assuming that the clock of the receiver is loosely synchronized with the sender. If the MAC key is still secret, then the receiver buffers the message. In addition, each receiver also checks whether the disclosed key is correct: the receiver has to iteratively apply the hash function, which is used to generate the key chain, to the disclosed key until the most recent (already received) *correct* key is resulted. Afterwards, the receiver can check the correctness of MACs in the buffered messages that were sent during interval I_i . Note that if intermediate keys are lost, they can be regenerated using later keys based on the property of one-way hash chains. Thus, if some disclosed keys are lost due to malicious packet dropping or jamming, a receiver can recover the key from keys disclosed later and check whether earlier messages are authentic.

μ Tesla is inappropriate for real-time applications, when messages should be authenticated immediately. For instance, alarms should be typically disseminated and authenticated with minimal delays. Moreover, if such alarms occur infrequently, a receiver may need to compute a long part of the hash chain to verify the authenticity of the key (in the case of infrequent messages, the sender releases keys that can be far away from

each other in the key chain). This can be exploited by the adversary to mount a DoS attack: the adversary sends incorrect keys to the receiver which performs key verification on the incorrect key and becomes overloaded (i.e., the receiver may need to perform thousands of hash computations which takes several seconds before it notices that the key is incorrect).

To circumvent the problem of authentication delay, the authors in [51] proposed a modified version of μ Tesla called RPT (Regular-Predictable Tesla). This protocol is an alternative of the μ Tesla for scenarios where the messages are sent at regular and predictable times. For instance, if we consider a monitoring application where the base station distributes the sensing tasks once per day but the sensors must continuously report measurements, we generally cannot postpone the execution of current sensing tasks for a whole day. RPT solves this problem by performing message broadcast jointly with key disclosure only after the distribution of the corresponding MAC (i.e., when the MAC is received by all nodes the sender can broadcast the message along with the corresponding secret keys). Of course, the delay between the MAC distribution and the key disclosure must be large enough to ensure that all receivers will receive the secret key by the time the sender releases the key. Keys are authenticated by hash chains similar to μ Tesla.

Previously, we assumed that the clocks of the sender and receiver nodes are synchronized off-line and the key-disclosure schedule along with the seed of the key-chain is pre-programmed into each node. However, μ Tesla was proposed to be used by symmetric key cryptography with a master key shared between the sender and each receiver. In this way, new receivers can also be bootstrapped; the receiver first sends a request to the sender, which then replies a message containing the current time for synchronization purposes, a key of the key chain used in a passed interval I_i , the start time of I_i , and the interval duration T_{int} along with the disclosure lag d . However, all these solutions does not scale to large networks with thousands of receiver nodes and may require an existing routing infrastructure. For this reason, several multi-level μ Tesla schemes were proposed in [49] where the initial parameters of the μ Tesla scheme can be effectively distributed in large sensor networks. Additionally, multi-level μ Tesla schemes can be used over a longer time period than basic μ Tesla scheme. Furthermore, basic μ Tesla has also been extended to multiple senders in [50], where the revocation of compromised senders is also treated. More interested readers are referred to [50, 49].

3.3.4 One-time signatures

A one-time signature is considerably faster to generate and verify than previously described digital signatures, but a private key can be used to sign only a single message. If a private key is used to sign multiple messages the probability that the adversary can successfully forge a valid signature can also significantly increases. Due to its lower computation demand, one-time signatures would be more suitable for sensor networks than RSA or ECDSA signature schemes. However, one-time signatures are still quite large compared to the message size. For instance, signing 16 bytes results in a 280 bytes long signature using the Merkle-Winternitz signature scheme [55] with 8 byte long hash chain values that is still too long in sensor networks. One-time signatures are only beneficial if we use short messages, or more precisely, messages with lower entropy, typically containing only a small number of bits (e.g., a 8 bytes long message has a 32 bytes long signature using the aforementioned scheme and this can be further reduced to 24 bytes if we require a bit more verification effort).

The problem still remains: how can we authenticate more than one messages without degrading security? In particular, the sender should provide a unique public key per message for the receivers. One solution is to pre-deploy n public keys at bootstrap and these keys can be used to sign the first n messages. Afterwards, the sender employing a μ Tesla scheme can distribute further public keys, which number is further denoted by k , at regular time in a periodic manner. In that case, the sender is limited to sign k messages between the μ Tesla key disclosure times. This solution entails a new problem: the greater is k the more memory resource the receiver needs. LEA (Low-Entropy Authentication) proposed in [51] resolves this resource problem by chaining all public keys, where the verification of one signature will automatically authenticate the public key of the next signature. More interested readers are referred to [51].

In summary, one-time signatures combined with some variant of the μ Tesla scheme can be used to authenticate broadcast messages in an effective way, but this solution is only practical for small sized messages due to the increased signature size.

3.4 Multi-path routing (to achieve robustness)

Multi-path routing, which encompasses delivering of data packets on multiple paths towards the destination, is a common technique to achieve robustness and load-balancing in every communication network. The multiple paths between the source and the destination can be partially or completely disjoint and they are maintained at the expense of increased energy consumption and traffic generation. Apart from load-balancing and robustness against node failures, multi-path routing also inherently provides some defense against selective forwarding attacks and malicious control packet dropping; in order to prevent a packet to reach the base station, the adversary must control a node on each used path to drop the packet. However, the more the number of (disjoint) paths is the less likely that the adversary can drop packets on all paths. We note that there are alternative methods to defend against malicious packet dropping in ad-hoc networks. However, they offer a limited usage in sensor networks, as they either induce heavy communication overload for sensor nodes by using promiscuous mode [53], or they could only be used with source routing protocols that are less attractive in sensor networks due to the increased length of the packet header [4]. Below, we review some multipath techniques used in sensor networks. They are further divided into three groups in order to ease their short exploration:

- The source makes multiple copies of a packet, and routes these copies on different paths in order to increase robustness [35, 28]. These paths can be calculated in advance and maintained proactively by sending data packets at a low rate *only* on these paths [28]. Alternatively, if the sources have data to send, they flood the *whole* network with data packets at a low rate, and the destination select the best quality paths according to some network metric [35]. In [28], two further localized methods were proposed to build disjoint multipaths and braided (partly disjoint) multipaths.
- The source routes the single copy of each packet on different paths per packet, where the paths are selected in a probabilistic or deterministic fashion in order to aid load-balancing, and thus prolong network-lifetime. In this category, centralized and decentralized approaches can be further distinguished. A centralized and probabilistic method was proposed in [71], where the probability that the packet is forwarded to a particular next-hop depends on the residual energy of the next-hop or the energy consumed by passing the packet to that next-hop. This calculation is done by each node independently from each other. In contrast to this, a centralized and deterministic approach was proposed in [47], where the paths are calculated by the base station based on the energy consumption of the *entire* path considering the residual energy of *all* nodes on that path.
- The source splits the original data packet into subpackets, adds some redundancy to each subpacket, and then sends each subpacket on one of the n available paths. As it was studied in [24], if some forward error correcting code is applied that corrects k ($k < n$) errors, then the method is a kind of trade-off between amount of traffic and reliability: even if some of the subpackets were lost, the original message can still be reconstructed due to the added redundancy to each subpacket (i.e., only k subpackets are needed at the destination to reconstruct the original message). In other words, even if the adversary manages to drop $n - k$ subpackets, the packet can still be reconstructed at the base station.

4 Summary

The security of routing protocols is a fundamental issue in sensor networks. Due to the limitations of the tiny sensor devices, the attacks mounted against the routing service can have devastating effects in wireless sensor networks. Moreover, some of the standard techniques that have been used to defend against these typical routing attacks so far are no longer desirable in sensor networks because of their high resource demands. Hence, designing secure routing protocols in wireless sensor networks is a challenging task.

In this part, we gave an up-to-date picture of the security problems of routing protocols in sensor networks. In particular, we first described the sensor routing specific adversary model with its primary objectives, and we also listed some possible attack methods used by this adversary to achieve her objectives. We demonstrated the severity of these attacks on real protocols: we showed how the adversary can subvert TinyOS beaconing, GPCR and Directed Diffusion along with simple attack scenarios. Then, we presented some countermeasures against the described attacks; we focused on link-layer security, secure neighbor discovery, broadcast authentication,

and robust data delivery. In the second part of the chapter, we illustrated how these efficient defense mechanisms can be used by the secure sensor network routing protocols.

Although there is an on-going research effort on designing new secure routing protocols for wireless sensor networks, the number of *secure* sensor network routing protocols is still limited. Additionally, the security analysis of these protocols has been done by only informal reasoning so far, however, precise and mathematically rigorous analysis would also be needed. We try to alleviate these problems in the rest of this work.

Part II

A formal framework for provably secure routing in WSNs

In this part, we present a formal model, in which security of routing can be precisely defined, and which can serve as the basis for rigorous security analysis of routing protocols proposed for wireless sensor networks. Our model is based on the simulation paradigm, where 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 he must run in time polynomial). This is a standard approach for defining security, however, it must be adopted carefully to the specific environment of wireless sensor networks.

First of all, we develop an adversary model that is different from the standard Dolev-Yao model (where the adversary can control all communications in the system). In wireless 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. In addition, we must also model the broadcast nature of radio communications.

Second, since we are aiming at designing secure routing protocols for wireless sensor networks, we must take into account that there are some attacks that exploit the constraint energy supply of sensor nodes (e.g., the adversary decreases the network lifetime by diverting the traffic in order to overload, and thus, deplete some sensor nodes). Hence, we explicitly model the energy consumption required to send messages on the wireless links of the network.

Third, we must also be careful with the definition of the output of the real-world model. It is tempting to consider the state stored in the routing tables of the nodes as the output, but an adversary can distort that state in unavoidable ways. This means that if we based our definition of security on the indistinguishability of the routing states in the dynamic model, then no routing protocol would satisfy that definition. Hence, we define the output of the model as a *suitable* function of the routing state, which hides the unavoidable distortions in the states. This function may be different for different types of routing protocols, but the general approach of comparing the output of this function in the real-world model to a specified „ideal” value remain the same. 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 the same in the real-world model in every simulation run, the protocol would still be secure if the case when the average length of the shortest paths is „incorrect” (the protocol is successfully attacked) occurs only with a negligible probability.

The rest of this part is organized as follows: We first present our adversary model adopted to wireless sensor networks, then we describe the dynamic model, and we give a general definition of routing security. Then, we illustrate the usage of our model in two ways. First, we represent a known insecurity of an authenticated version of the TinyOS routing protocol in the model. Second, we prove that INSENS [20] is a secure link-state routing protocol in our model.

5 The model

5.1 Adversary model

Based on Section 2.1, we assume that the adversary can capture honest sensor nodes in our model, and she may be able to compromise their cryptographic secrets (assuming that such secrets are used in the system). Thus, we assume in our model that the adversary can compromise cryptographic material (i.e., our adversary is an *insider adversary* in this sense). In addition, all adversarial nodes may be able to communicate in 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.

5.2 Network model

We assume that each honest device has exactly one antenna in the network. If the adversary uses several antennas we represent each of them by a distinct node. The network nodes are considered to be static, and we further assume that there is a single base station in the network.

The honest nodes in the network are denoted by v_0, \dots, v_k , where v_0 denotes the base station, and adversarial nodes are denoted by v_{k+1}, \dots, 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, we introduce a matrix \mathbf{E} , called *reachability matrix*, with size $n \times n$. Here, $\underline{E}_{i,j}$ ($0 \leq i, j \leq n - 1$) represents the energy level needed for v_i to communicate with v_j (i.e., if node v_i uses energy level $\underline{E}_{i,j}$ to broadcast a message, then v_j also receives the message).

We 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). We denote the set of these identifiers by L , and there is a function $\mathcal{L} : V \rightarrow L \cup \{\text{undef}\}$ that assigns an identifier to each node, where $\text{undef} \notin L$. According to our adversary model, we assume that the adversary has (authenticated) identifier(s) in the network, denoted by L^* that can be used by all adversarial nodes, i.e., $\mathcal{L}(v_j^*) = L^*$ for all $1 \leq j \leq m$.

Since adversarial nodes can communicate via out-of-band channels, we merge each adversarial node into a single adversarial node. Accordingly, we model the modified connectivity by matrix \mathbf{E}^* , called *reduced reachability matrix*. \mathbf{E}^* can be unambiguously derived from \mathbf{E} with size $(k+2) \times (k+2)$ in the following way. For all i, j ($0 \leq i, j \leq k$), $\underline{E}_{i,j}^*$ is identical to $\underline{E}_{i,j}$. For an honest node v_ℓ ($0 \leq \ell \leq k$), $\underline{E}_{\ell,k+1}$ represents the minimal energy level that is needed for v_ℓ to communicate with at least one adversarial node. Similarly, $\underline{E}_{k+1,\ell}$ represents the minimal energy level that is needed for the adversary to communicate with v_ℓ (i.e., there exists at least one adversarial node that can communicate with v_ℓ using energy level $\underline{E}_{k+1,\ell}$).

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

Configuration: A *configuration* of a network is a quintuple $\text{conf} = (V, V^*, \mathcal{L}, \mathbf{E}, \mathcal{C})$, 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.

5.3 Security objective function

Diverse sensor 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 sensor routing protocols use a trade-off to satisfy conflicting requirements [73, 47].

This small argument 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. By the above argument, a routing protocol that is secure against attacks aiming at decreasing network-lifetime cannot be secure against attacks aiming at increasing network delay. We model the negatively correlated requirements of routing, and essentially, our security objectives in a very general manner. We represent the output of a routing protocol, which is actually the ensemble of the routing entries of the honest nodes, with a given configuration conf by a matrix $\underline{T}^{\text{conf}}$ with size $k+1 \times k+1$.¹ $\underline{T}_{i,j}^{\text{conf}} = 1$, if honest node v_i sends every message to an honest node identified by $\mathcal{L}(v_j)$ in order to deliver the message to the base station, otherwise let $\underline{T}_{i,j}^{\text{conf}}$ be 0. In the rest of this work, we shortly refer to the result of a routing protocol with a given configuration as a *routing topology*, which can be considered as a directed graph described by matrix $\underline{T}^{\text{conf}}$. In the following, we will omit the index conf of \underline{T} when the configuration can be unambiguously determined in a given context. In fact,

¹Of course, here we only consider the result of the protocol with respect to the honest nodes, since the adversarial nodes may not follow the protocol rules faithfully.

\underline{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.

Let us denote the set of all configurations by \mathbb{G} . Furthermore, \mathbb{T} denotes the set of the routing topologies of all configurations. The security objective function $\mathcal{F} : \mathbb{G} \times \mathbb{T} \rightarrow \mathbb{R}$ 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. We note that the definition of \mathcal{F} is protocol dependent. 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

$$\mathcal{F}(conf, \underline{T}^{conf}) = \frac{1}{k} \sum_{i=1}^k \mathcal{E}(v_i, conf, \underline{T}^{conf})$$

where $\mathcal{E} : V \times \mathbb{G} \times \mathbb{T} \rightarrow \mathbb{R}$ 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. Since \underline{T}^{conf} is a random variable, the output of \mathcal{F} is a random variable too. If the distribution of this output in the presence of an attacker non-negligibly differs from the distribution when there’s no attacker, then the protocol is not secure. If we intend to compare routing trees based on network delay a simple security objective function may be

$$\mathcal{F}(conf, \underline{T}^{conf}) = \frac{1}{k} \sum_{i=1}^k \mathcal{M}(v_i, conf, \underline{T}^{conf})$$

where $\mathcal{M} : V \times \mathbb{G} \times \mathbb{T} \rightarrow \mathbb{R}$ assigns the length of the path from a node to v_0 in a routing topology of a configuration.

In the rest of the paper, we assume that \mathcal{F} returns 1 if the routing topology is correct. Otherwise, it returns 0.

5.4 Dynamic model

The dynamic model is based on the simulation paradigm and similar to [2, 97]. However, our model deviates from these works in the sense that we do not distinguish a real-world model and an ideal-world model as usual in the simulation paradigm, but for the simplicity of the presentation, we define a single model that represents the real operation of the network. and contains an adversary. This real-world adversary is not constrained apart from requiring it to run in time polynomial. 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 resulting routing topology) in order to decide whether the protocol functions correctly or not. Once the model is defined, the goal is to prove that for any real-world adversary, the probability that the security objective function is not satisfied is negligible.

The real-world model that corresponds to a configuration $conf = (V, V^*, \mathcal{L}, \underline{E}, C)$ and adversary \mathcal{A} is denoted by $sys_{conf, \mathcal{A}}$, and it is illustrated on Figure 6. We model the operation of the protocol participants by interactive and probabilistic Turing machines. Correspondingly, we represent the adversary, the honest sensor 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, we 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 \underline{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 energy 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:

- (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 $cont$, energy 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.
- (DEL, ℓ_{tar}, e): Special DEL message, that is sent by the adversary to machine C , models the modification capability of the adversary. When receiving a message DEL with identifier $\ell_{tar} \in L$, machine C does not deliver any messages sent by 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 that we described in Subsection 5.1.

In a more formal way, when reading a message $msg_{in}^* = (MSG, \ell_{sndr}, cont, e, dest)$ 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 $v' \in V_e$ $e_{v_j, v'} \leq e$. 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
 - $msg_{out}^* = (MSG, \ell_{sndr}, cont, dest)$ to the input tapes of machines corresponding to adversarial nodes in $V_e \setminus \{v_j^*\}$
2. otherwise C discards msg_{in}^*

When reading a message $msg_{in}^* = (JAM, e)$ from out_j^* , C determines the set of nodes which receive the message by calculating $V_e \subseteq V$, such that for all $v' \in V_e$ $e_{v_j, v'} \leq e$. 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 $msg_{in}^* = (DEL, \ell_{tar}, e)$ from out_j^* , C determines the set of nodes which receive the message by calculating $V_e \subseteq V$, such that for all $v' \in V_e$ $e_{v_j, v'} \leq e$. 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 $V_e \setminus \{v_j^*\}$
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 $v' \in V_e$ $e_{v_j, v'} \leq e$. 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 $V_e \setminus \{v_i\}$
 - $msg_{out}^* = (MSG, \ell_{sndr}, cont, dest)$ 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.
- 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 deploy several antennas in the network field, which are connected to a central adversary logic. In this convention, node v_j^* corresponds to an adversarial antenna, 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 $msg_{in}^* = (MSG, \ell_{sndr}, cont, e, dest)$, where $dest \in L \cup \{*\}$.

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

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

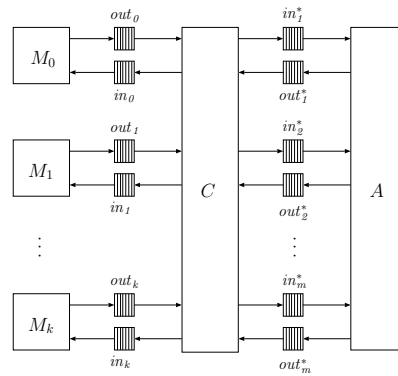


Figure 6: The real-world model.

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 routing topology defined in Subsection 5.3 and configuration $conf$. The routing topology is represented by the ensemble of the routing entries of machines M_i . We 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.

5.5 Definition of secure routing

We 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, we define routing security as follows:

Definition 1 A routing protocol is secure with security objective function \mathcal{F} , if for any configuration $conf$ and any adversary \mathcal{A} , the probability that $Out_{conf, \mathcal{A}}^{\mathcal{F}}$ equals to zero is a negligible function of κ .²

²a 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 $x > N_c$), $\mu(x) \leq x^{-c}$

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 .

6 Example 1: Insecurity of TinyOS routing

In this section, we present an authenticated routing mechanism based on the well-known TinyOS routing, and we show that it is not secure in our model for a given security objective function representing a very minimal security requirement.

6.1 Operation of an authenticated routing protocol

Originally, the authors of TinyOS implemented a very simple routing protocol, where each node uses a globally unique identifier. The base station periodically initiates a routing topology discovery by flooding the network by a beacon message. Upon reception of the first beacon within a single beaconing interval, each sensor node stores the identifier of the node, from which it received the beacon, as its parent (aka. next-hop towards the base station), and then re-broadcasts the beacon after changing the sender identifier to its own identifier. As for each node only one parent is stored, the resulted routing topology is a tree. Every sensor node receiving a data packet forwards that towards the base station by sending the packet to its parent. A lightweight cryptographic extension is employed in [62] in order to authenticate the beacon by the base station. This authenticated variant of TinyOS routing uses μ Tesla scheme to provide integrity for the beacon; each key is disclosed by the next beacon in the subsequent beaconing interval. We remark that this protocol has only been defined informally that inspired us to present a new protocol, which provides the "same" security as the authenticated routing protocol in [62], but due to its simplicity it fits more in demonstrating the usage of our model. Consequently, the presented attack against this new protocol also works against the protocol in [62]. We must note again that this protocol is only intended to present the usefulness of our model rather than to be considered as a proposal of a new sensor routing protocol.

We 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} as well as to store some beacons in its internal memory. We 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:

$$B \rightarrow * : msg_1 = (\text{BEACON}, rnd, Id_B, sig_B)$$

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, stores msg_1 in its internal memory, and re-broadcasts the beacon by changing the sender identifier Id_B to its own identifier Id_X :

$$X \rightarrow * : msg_2 = (\text{BEACON}, rnd, Id_X, sig_B)$$

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 .

In the rest, we shortly refer to this protocol as ABEM (Authenticated Beaconing Mechanism).

6.2 Formalization of a simple attack

A simple security objective is 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 have an adversarial neighboring node $v_{\ell_1}^*$ and $v_{\ell_2}^*$, resp., such that $E_{i,k+\ell_1}$ and $E_{k+\ell_2,j}$ are finite values, where $1 \leq \ell_1, \ell_2 \leq m$ and $\ell_1 \neq \ell_2$ may hold.

In order to formalize this minimal security requirement, we introduce the following security objective function

$$\mathcal{F}^{\text{ABEM}}(\text{conf}, \underline{T}) = \begin{cases} 1, & \text{if } \forall i, j : \underline{T}_{i,j} \cdot \underline{E}'_{i,j} \cdot (\prod_{\ell=1}^m \underline{E}'_{i,k+\ell} + \prod_{\ell=1}^m \underline{E}'_{k+\ell,j}) = 0 \\ 0, & \text{otherwise} \end{cases}$$

where we derive matrix \underline{E}' with size $n \times n$ from \underline{E} , so that $\underline{E}'_{i,j} = 1$, if $\underline{E}_{i,j} = \infty$, otherwise $\underline{E}'_{i,j} = 0$. In other words, $\underline{E}'_{i,j} = 1$, if v_i cannot send a message directly to v_j , otherwise $\underline{E}'_{i,j} = 0$.

We will show that ABEM is not secure in our model for security objective function $\mathcal{F}^{\text{ABEM}}$. In particular, we present a configuration conf and an adversary \mathcal{A} , for which the probability that $\text{Out}_{\text{conf}, \mathcal{A}}^{\mathcal{F}^{\text{ABEM}}}$ equals to zero is a non-negligible function of the key-size of the signature scheme. Equivalently, we show that for a real-world adversary \mathcal{A} , $\mathcal{F}^{\text{ABEM}}(\text{conf}, \underline{T}) = 0$ with a probability that is a non-negligible function of κ . Moreover, the success probability of the real-world adversary \mathcal{A} described below is independent from κ .

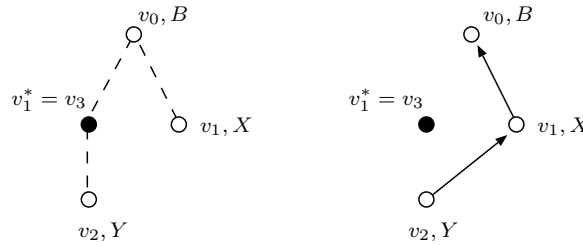


Figure 7: A simple attack against ABEM. v_0 , v_1 , and v_2 are honest nodes with identifiers $\mathcal{L}(v_0) = B$, $\mathcal{L}(v_1) = X$, and $\mathcal{L}(v_2) = Y$, whereas v_1^* is an adversarial node. $\underline{E}_{1,0}$, $\underline{E}_{3,0}$, $\underline{E}_{2,3}$ are finite values, and $\underline{E}_{3,1} = \underline{E}_{2,0} = \underline{E}_{2,1} = \infty$. Links are assumed to be symmetric, i.e., $\underline{E}_{i,j} = \underline{E}_{j,i}$. The configuration is illustrated on the left-hand side, where a dashed line denotes a direct link. In the routing topology of the real-world model, on the right-hand side, v_2 sets X as its parent identifier, however, $\underline{E}_{2,1} = \infty$ and $\underline{E}_{3,1} = \infty$.

The configuration conf and the result of the attack is depicted on Figure 7. We assume that the base station broadcasts only a single beacon during the computational process, i.e., only a single beaconing interval is analyzed in our model. At the beginning, the base station B floods the network by a beacon

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

Both adversarial node v_1^* 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:

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

In parallel, v_1^* modifies the beacon by replacing sender identifier B to X , and broadcasts the resulted beacon:

$$v_1^* \rightarrow * : \text{msg}'_2 = (\text{BEACON}, \text{rnd}, X, \text{sig}_B)$$

Upon the reception of msg'_2 , node Y sets X as its parent, since sig_B is correct.

In the real-world model, these actions result $\underline{T}_{2,1} = 1$, which implies that $\mathcal{F}^{\text{ABEM}}(\text{conf}, \underline{T}) = 0$. On the contrary, $\mathcal{F}^{\text{ABEM}}(\text{conf}, \underline{T}')$ never equals to 0, where \underline{T}' represents the routing topology in the ideal-world

model. Let us assume that $\mathcal{F}^{\text{ABEM}}(\text{conf}, \underline{T}') = 0$, which means that $\underline{T}'_{1,2} = 1$ or $\underline{T}'_{2,1} = 1$. $\underline{T}'_{1,2} = 1$ is only possible, if X receives

$$\text{msg}'_3 = (\text{BEACON}, \text{rnd}, Y, \text{sig}_B)$$

However, it yields contradiction, since $\underline{E}_{3,1} = E_{2,1} = \infty$, and B never broadcasts msg'_3 . Similarly, if $\underline{T}'_{2,1} = 1$ then Y must receive msg'_2 , which means that v_1^* must broadcast msg'_2 . Conversely, B never broadcasts msg'_2 , and $\underline{E}_{3,1} = \infty$. Therefore, v_1^* can only broadcast msg'_2 , if he successfully modifies msg'_1 or forges msg'_2 . However, it also contradicts our assumption that the ideal-world adversary cannot modify and inject messages in the ideal-world model.

7 Example 2: Security of INSENS

In our second example, we further demonstrate the strength of our model by showing that INSENS [20] is a provably secure link-state routing protocol in our model.

7.1 Operation of INSENS

In this subsection, we describe the operation of INSENS (for more detailed description, see [20]). In this paper, we are only concerned with the topology (route) discovery mechanism of INSENS and not with the data forwarding mechanism.

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:

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

where REQ is a constant message type identifier, hash is the next element of the hash chain in reversed direction, and v_0 identifies the base station. The hash chain mechanism is intended to provide authenticity and some defense against 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

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

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 $\text{MAC}_{v_{\ell_{i-1}}}^{\text{REQ}}$ in conjunction with $\mathcal{L}(v_{\ell_{i-1}})$ locally. Before re-broadcasting, v_{ℓ_i} replaces $\text{MAC}_{v_{\ell_{i-1}}}^{\text{REQ}}$ in the request to $\text{MAC}_{v_{\ell_i}}^{\text{REQ}}$, which is the MAC generated by v_{ℓ_i} on list $[v_0, \dots, v_{\ell_{i-1}}, v_{\ell_i}]$, REQ, and hash using the symmetric key shared with v_0 . Finally, v_{ℓ_i} re-broadcasts the following request:

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

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:

$$v_{\ell_x} \rightarrow v_{\ell_{x-1}} : (\text{NLIST}, \text{hash}, \text{MAC}_{v_{\ell_{x-1}}}^{\text{REQ}}, v_{\ell_x}, \text{Enc}_{v_{\ell_x}}(\text{path}_{v_{\ell_x}}, \text{neighborlist}_{v_{\ell_x}}), \text{MAC}_{v_{\ell_x}}^{\text{NLIST}})$$

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; $\text{MAC}_{v_{\ell_{x-1}}}^{\text{REQ}}$ is the MAC, called parent MAC³, of $v_{\ell_{x-1}}$ sent in the corresponding request; v_{ℓ_x} is the identifier of the message originator; $\text{Enc}_{v_{\ell_x}}(\text{path}_{v_{\ell_x}}, \text{neighborlist}_{v_{\ell_x}})$ is the neighborhood information and the path information of v_{ℓ_x} encrypted by the symmetric key shared with the base station; $\text{neighborlist}_{v_{\ell_x}}$ contains the identifiers of each neighboring node *and* their corresponding MACs received in Phase 1; $\text{path}_{v_{\ell_x}}$ is $[v_{\ell_x}, \dots, v_{\ell_1}, v_0, \text{MAC}_{v_{\ell_x}}^{\text{REQ}}]$, which is the reverse of the path received in the

³In this context, parent node is the next-hop that forwards neighborhood information, and not measured data, towards the base station.

corresponding request message including the MAC of node v_x ; and finally $\text{MAC}_{v_{\ell_x}}^{\text{NLIST}}$ is the MAC computed by node v_{ℓ_x} on NLIST, hash, $\text{path}_{v_{\ell_x}}$, and $\text{neighborlist}_{v_{\ell_x}}$.

A node receiving the reply message first checks if the node is the parent of the sender (i.e., $\text{MAC}_{v_{\ell_x-1}}^{\text{REQ}}$ message equals to its own MAC that has been broadcast with request containing hash). Then, the node replaces the parent MAC in the message to 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

$$(\text{NLIST}, \text{hash}, v_{\ell_x}, \text{Enc}_{v_{\ell_x}}(\text{path}_{v_{\ell_x}}, \text{neighborlist}_{v_{\ell_x}}), \text{MAC}_{v_{\ell_x}}^{\text{NLIST}})$$

the base station checks whether all the MACs are correct, after decrypting $\text{Enc}_{v_{\ell_x}}(\text{path}_{v_{\ell_x}}, \text{neighborlist}_{v_{\ell_x}})$ ⁴. If all verifications are successful, the base station computes the forwarding table for each node using a global centralized algorithm detailed in [20].

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} :

$$v_0 \rightarrow v_{\ell_1} : (\text{FTABLE}, v_{\ell_1}, \text{hash}, \text{Enc}_{v_{\ell_1}}(\text{fable}_{v_{\ell_1}}), \text{MAC}_{v_{\ell_1}}^{\text{FTABLE}})$$

where FTABLE is a constant message type identifier, $\text{Enc}_{v_{\ell_1}}(\text{fable}_{v_{\ell_1}})$ is the encrypted form of the forwarding table of v_{ℓ_1} , and $\text{MAC}_{v_{\ell_1}}^{\text{FTABLE}}$ 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 $\text{fable}_{v_{\ell_1}}$, if $\text{MAC}_{v_{\ell_1}}^{\text{FTABLE}}$ is correct.

7.2 Security analysis

In this subsection we show that INSENS described in Section 7.1 is secure in our model. We 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 can reach node v_j by either a direct contact, or an adversarial relaying (one can also imagine the adversarial relaying as a wormhole between some honest nodes).

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 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 $(k+2) \times (k+2)$, denoted by \mathbf{N} , describes the neighborhood relations among the sensor nodes that is believed by the base station to be correct (i.e., $\underline{N}_{i,j} = 1$ if the base station believes that v_i is a neighbor of v_j , otherwise $\underline{N}_{i,j} = 0$). The output of \mathcal{G} is the ensemble of the routing entries (the routing topology) that should be set by each node.

Now, we prove that INSENS is secure with respect to the aforementioned security objective.

Theorem 1 *Let us consider the following security objective function:*

$$\mathcal{F}(\text{conf}, \mathbf{T}) = \begin{cases} 1, & \text{there exists } \mathbf{E}' \text{ such that for all } i, j \text{ it holds that if } \underline{T}_{i,j} = 1, \text{ then } \mathcal{G}(\mathbf{E}')_{i,j} = 1 \\ 0, & \text{otherwise} \end{cases}$$

⁴Actually, the MACs in the $\text{neighborlist}_{v_{\ell_x}}$ can only be checked when the NLIST messages of the corresponding nodes in $\text{neighborlist}_{v_{\ell_x}}$ are also received.

where \mathbf{E}' with size $(k+2) \times (k+2)$ is derived from \mathbf{E}^* , such that $\underline{E}'_{i,j} = 0$, if $\underline{E}^*_{i,j} = \infty$, and $\underline{E}'_{i,k+1} = \infty$ or $\underline{E}'_{k+1,j} = \infty^5$. INSENS is secure with respect to \mathcal{F} , if the MAC scheme is secure against existential forgery, and the symmetric encryption scheme is secure against plaintext recovery attack.

Proof We 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 $\underline{T}_{i,j} = 1$ and there does not exist any \mathbf{E}' , derived from \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, we introduce the following events:

- (i) $C_1^{i,j}$ denotes the event that $\underline{T}_{i,j} = 1$, but $\mathcal{G}(\mathbf{N})_{i,j} = 0$,
- (ii) $C_2^{i,j}$ denotes the event that $\underline{T}_{i,j} = 1$, $\mathcal{G}(\mathbf{N})_{i,j} = 1$, and $\underline{N}_{i,j} = 1$, but $\underline{E}_{i,j}^* = \infty$ as well as $\underline{E}_{i,k+1}^* = \infty$ or $\underline{E}_{k+1,j}^* = \infty$.

We 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 $P^{\mathcal{A}}$:

$$P^{\mathcal{A}} \leq \sum_{\forall i,j:i \neq j, i \neq 0} \mathbf{P}\left(C_1^{i,j}\right) + \sum_{\forall i,j:i \neq j, i \neq 0} \mathbf{P}\left(C_2^{i,j}\right)$$

We show that $\mathbf{P}\left(C_1^{i,j}\right)$ is a negligible function of κ_1 , and $\mathbf{P}\left(C_2^{i,j}\right)$ 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 $\mathbf{P}\left(C_1^{i,j}\right)$: If $C_1^{i,j}$ occurs, then M_i receives an FTABLE message, which contains the routing information of node v_i :

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

v_i infers from ftable'_{v_i} that $\underline{T}_{i,j} = 1$, since $\text{MAC}'_{v_i}{}^{\text{FTABLE}}$ is a correct MAC. We show that it is only possible if $\text{MAC}'_{v_i}{}^{\text{FTABLE}}$ is a successfully forged MAC by \mathcal{A} .

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

Consequently, $C_1^{i,j}$ 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 $\mathbf{P}\left(C_2^{i,j}\right)$: If $C_2^{i,j}$ occurs, then M_0 receives an NLIST message, which contains the neighborhood information of node v_j :

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

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

1. $\text{MAC}'_{v_j}{}^{\text{NLIST}}$ is a successfully forged MAC by \mathcal{A} , if v_j is an honest node.
2. There exists a node v_t ($1 \leq t \leq k$), for which $\underline{E}_{i,t}^* < \infty$ and \mathcal{A} successfully recovered the plaintext from $\text{Enc}_{v_t}(\text{path}_{v_t}, \text{neighborlist}_{v_t})$ that is sent in the corresponding NLIST message by v_t .
3. $\text{MAC}'_{v_i}{}^{\text{REQ}}$ that is received by v_j is a successfully forged MAC by \mathcal{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.

⁵The rationale behind the definition of \mathbf{E}' is that the adversary can always drop messages that should be tolerated. However, we can defend against illegal injection and modification of messages by using appropriate cryptographic primitives.

- (i) Based on the argument of the negligibility of $C_1^{i,j}$, we know that $MAC_{v_j}^{NLIST}$ can only be generated by M_j . Thus, M_j received a REQ message denoted by

$$msg' = (\text{REQ}, \text{hash}, [v_0, \dots, v_i], MAC_{v_i}^{\text{REQ}})$$

We know that msg' is never relayed by machines $M_0, \dots, M_{i-1}, M_{i+1}, \dots, M_k$, since these machines never send any REQ messages containing a path where the last element is v_i (such as path $[v_0, \dots, v_i]$ in msg'). Therefore, M_j receives msg' from A implying that $\underline{E}_{k+1,j}^* < \infty$.

Since v_i is not an adversarial node, $MAC_{v_i}^{\text{REQ}}$ cannot be generated by machines $M_0, \dots, M_{i-1}, M_{i+1}, \dots, M_k, A$. Therefore, only M_i can generate $MAC_{v_i}^{\text{REQ}}$. We know that msg' cannot be sent to M_j by M_i , since $\underline{E}_{i,j} = \infty$. We will show that $\underline{E}_{i,k+1}^* < \infty$, which is a contradiction.

First, let us assume that $\underline{E}_{i,k+1}^* = \infty$. In order to construct msg' , A can only infer $MAC_{v_i}^{\text{REQ}}$ 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 $MAC_{v_i}^{\text{REQ}}$ in an encrypted form. In that case, $MAC_{v_i}^{\text{REQ}}$ must be inferred from $\text{Enc}_{v_t}(\text{path}_{v_t}, \text{neighborlist}_{v_t})$, which contradicts to our assumption. Therefore, $\underline{E}_{i,k+1}^* < \infty$.

- (ii) Let us assume that $\underline{E}_{i,j}^* = \infty$, where $j = k + 1$. Similar to case (i), A can only infer $MAC_{v_i}^{\text{REQ}}$ from the messages sent by the neighbors of v_i , as A is unable to forge $MAC_{v_i}^{\text{REQ}}$. Thus, A must recover $MAC_{v_i}^{\text{REQ}}$ from encrypted neighborlists. However, by assumption, the adversary cannot do this. This means that $\underline{E}_{i,j}^* < \infty$, which is a contradiction again.

Consequently, $C_2^{i,j}$ 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. ■

7.3 Discussion

We 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 MAC^{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 7.2, as the adversary could cause the base station to consider false neighborhood relations. Furthermore, as MAC^{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 [20], where the authors used informal reasoning to prove the security of INSENS.

In contrast to this, our formal security analysis would reveal such flaw in a routing protocol: if encryption had not been employed, we could not have claimed in the proof that the adversary can retrieve the MAC^{REQ} of a non-neighbor node only from the encrypted neighborlist of other nodes. Therefore, our formal analysis lead us 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.

8 Summary

In this part, we described a formal security model for routing protocols in wireless sensor networks. Our model is based on the well-known simulation paradigm, but it differs from previously proposed models in several important aspects. First of all, the adversary model is carefully adopted to the specific characteristics of wireless sensor networks. In our model, the adversary is not all-powerful, but it can only interfere with communications within its own radio range. A second important contribution is that we defined the output of the dynamic model as a suitable function of the routing state of the honest nodes, instead of just using the

routing state itself as the output. This allows us to model different types of routing protocols in a common framework. In addition, this approach hides the unavoidable distortions caused by the adversary in the routing state, and in this way, it makes our definition of routing security satisfiable.

We demonstrated the usefulness of our framework on two illustrative examples. First, we considered an authenticated version of the TinyOS beaconing routing protocol, and we showed how an attack against this protocol can be represented in our formal model. Second, we proved that INSENS, which is a secure sensor network routing protocol, is indeed secure in our model. Although INSENS is provably secure in our model, it is not scalable to large-scale networks due to its centralized nature, and the base station is a single point of failure. In the next part, we propose a novel secure sensor network routing protocol that does not have these unfavourable properties and considers the specifics of wireless sensor networks.

Part III

Secure-TinyLUNAR: A provably secure routing protocol for WSNs

Although there are some secure sensor network routing protocols in the literature, these are only applicable to specific sensor applications. Moreover, their security has been analyzed only by informal reasoning, which is an error-prone method. On the other hand, 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 [2]. An alternative solution could be to apply some secure ad hoc network routing protocol like [88] [68] [33]. 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 work, we 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 Codes (MAC) exclusively in the route discovery phase.

Secure-TinyLUNAR [93] is the secure variant of TinyLUNAR [59] which is a reactive routing protocol proposed for wireless sensor networks. TinyLUNAR is extensively used in the UbiSec&Sens research project due to its low communication and memory overhead. 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. Although TinyLUNAR has a slightly greater RAM consumption than other reactive routing protocols like tinyAODV [61], it uses considerably less 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, we will 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.

We also show that Secure-TinyLUNAR is a provably secure label switching routing protocol in our model described in Part 2. This model considers those attacks that aim to corrupt the routing entries of the nodes creating incorrect routing state. Our framework also has been successfully used so far to analyze the security of several multi-hop routing protocols like Ariadne [97], endairA [97], SRP [97], ARAN [95], SAODV [95] or INSENS [98]. We further demonstrate the strength of this framework by showing that Secure-TinyLUNAR is also provably secure. In particular, we first adapt this simulation-based model to secure label-switching routing, and prove that Secure-TinyLUNAR is indeed secure in that model. Then, we evaluate the performance of Secure-TinyLUNAR by TOSSIM simulations. First, we implement Secure-TinyLUNAR by extending TinyLUNAR with security mechanisms. Then, using this implementation we perform TOSSIM (PowerTOSSIM) simulations and compare the network delay, delivery ratio and energy consumption of Secure-TinyLUNAR to TinyLUNAR.

9 Insecurity of TinyLUNAR

In this section, we first give a brief overview of the operation of TinyLUNAR. One can read a more detailed description in [59]. Then, we show simple attacks against TinyLUNAR, whereby we motivate the development of Secure-TinyLUNAR.

9.1 Operation of TinyLUNAR

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

$$S \rightarrow * : (\text{RREQ}, \text{rnd}, S, D, \text{addr}_S, \text{label}_{S \rightarrow S}^{\text{In}})$$

where rnd is a randomly generated request id, $\text{label}_{S \rightarrow S}^{\text{In}}$ 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, $\text{label}_{S \rightarrow S}^{\text{In}}$ 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 been received the request earlier based on rnd , S , and D . If so, J silently drops the request. Otherwise, J stores the quadruple $(\text{addr}_S, \text{label}_{S \rightarrow S}^{\text{In}}, \text{rnd}, \text{lifetime})$ in its routing table, where lifetime is set to a predefined value MaxLifetime and addr_S is the local network address of the neighboring node from which the request is received. The value of 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 MaxLifetime . Using this entry, J can forward messages to S . Afterwards, J broadcasts the message as follows:

$$J \rightarrow * : (\text{RREQ}, \text{rnd}, S, D, \text{addr}_J, \text{label}_{J \rightarrow S}^{\text{In}})$$

where addr_J is the locally unique network address of J , and $\text{label}_{J \rightarrow S}^{\text{In}}$ is the incoming label of J towards S . Essentially, $\text{label}_{J \rightarrow S}^{\text{In}}$ is the local memory address of the routing entry where J stores the corresponding entry pointing to S (i.e., this entry contains the five-tuple S , addr_S , rnd , $\text{label}_{S \rightarrow S}^{\text{In}}$, and 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 is 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 receives the request. After that D sends a reply to S :

$$D \rightarrow Z : (\text{RREP}, \text{rnd}, \text{addr}_D, \text{label}_{Z \rightarrow S}^{\text{Out}}, \text{label}_{D \rightarrow D}^{\text{In}})$$

where rnd is the random identifier of the corresponding request originated from S , $\text{label}_{Z \rightarrow S}^{\text{Out}}$ is the incoming label of Z towards S (i.e., the outgoing label of D towards S) received in the request, and $\text{label}_{D \rightarrow D}^{\text{In}}$ is the incoming label of D . Here, $\text{label}_{D \rightarrow D}^{\text{In}}$ is a memory address inside the routing table of D and similar 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 $\text{label}_{D \rightarrow D}^{\text{In}}$, where addr_D is the local network address of the neighboring node from which the reply is received. From now, Z can forward messages to D . Then, Z looks up the entry addressed by $\text{label}_{Z \rightarrow S}^{\text{Out}}$ 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 :

$$Z \rightarrow K : (\text{RREP}, \text{rnd}, \text{addr}_Z, \text{label}_{K \rightarrow S}^{\text{Out}}, \text{label}_{Z \rightarrow D}^{\text{In}})$$

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 $\text{label}_{K \rightarrow S}^{\text{Out}}$.

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, from the time when 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 do 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. 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.

9.2 Attacks against TinyLUNAR

In this subsection, we gather the basic attacks against the route discovery process of TinyLUNAR. The main types of attacks include tunnelling, rushing, selective forwarding of control messages, replaying of control messages, Denial-of-Service attacks, the corruption of routing tables, and the disruption of neighbor discovery (see [94] for a more comprehensive overview).

In this paper, we consider those attacks that aim to corrupt the routing entries of honest nodes (i.e., the adversary causes honest nodes to have incorrect routing entries). An incorrect entry points to a node, which is not a neighbor, or points to a neighbor through which no packet can be delivered to the intended destination. In order to defend against the rest of the attacks, one can use the corresponding countermeasures [94].

In the following, we argue that impersonation attacks cause incorrect routing entries in TinyLUNAR. Thus, in Section 10.1, our first steps will be to defend against these impersonation attacks.

Source impersonation: The adversary can use any honest node identifier as the source identifier of any request messages. For instance, in Figure 8a, 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 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 8b 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 8c, 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 she 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 caught 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 .

In the following sections, we first describe the operation of Secure-TinyLUNAR. As a first step, we prevent the impersonation attacks that are described in Subsection 9.2. Secure-TinyLUNAR is the secure variant of TinyLUNAR, where we use pairwise message authentication codes (MACs) to authenticate routing messages between immediate neighbors and also to ensure source/destination authenticity. Finally, in Subsection 10.2.3, we show that this new protocol is provably secure in a model which is adapted to secure label-switching from [98].

10 Secure-TinyLUNAR

10.1 Operation of Secure-TinyLUNAR

We only discuss the main operational differences with respect to the original (and insecure) TinyLUNAR protocol. We assume that each pair of nodes share a symmetric pairwise key in the network. Any symmetric key

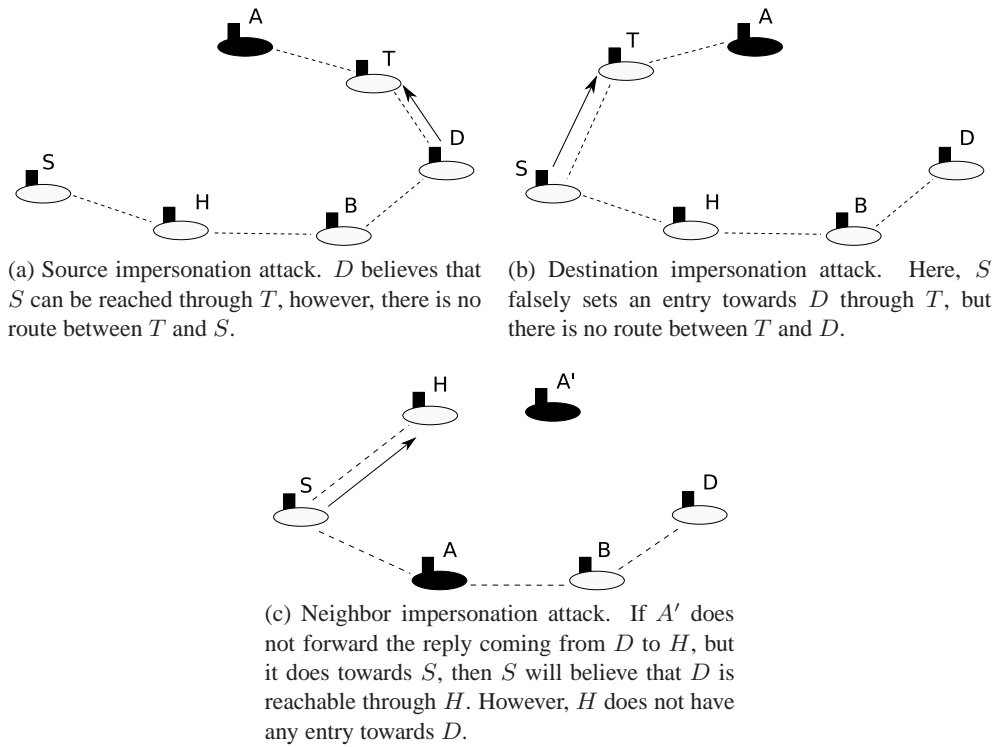


Figure 8: Impersonation attacks against TinyLUNAR. Dashed lines denote the neighborhood relations, whereas arrows denote the routing entries.

pre-distribution schemes proposed for wireless sensor networks (see [99] for a good overview) can be employed here. Additionally, it is also assumed that each node is aware of its local (one-hop) neighborhood.

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 unicasts the following route request message to *each* neighbor:

$$\text{for all } x, S \rightarrow N_x^S : (\text{RREQ}, \text{rnd}, S, D, \text{addr}_S, \text{label}_{S \rightarrow S}^{\text{In}}, \text{addr}_{N_x^S}, \text{MAC}_{S,D}, \text{MAC}_{S,N_x^S}^{\text{prv}})$$

where $\text{rnd}, S, D, \text{addr}_S, \text{label}_{S \rightarrow S}^{\text{In}}$ are the same as in the original TinyLUNAR protocol, $\text{addr}_{N_x^S}$ is the local network address of a neighbor N_x^S , $\text{MAC}_{S,D}$ is the message authentication code generated by S on the elements of the message excluding addr_S and $\text{label}_{S \rightarrow S}^{\text{In}}$ using the pairwise key shared with D . After generating $\text{MAC}_{S,D}$, S generates previous-hop MAC $\text{MAC}_{S,N_x^S}^{\text{prv}}$ on all elements of the message using the pairwise key shared with neighbor N_x^S . Upon the reception of this broadcast message, a neighboring node J checks the authenticity of the message by verifying $\text{MAC}_{S,J}^{\text{prv}}$. In case it is successful, node J removes $\text{MAC}_{S,J}^{\text{prv}}$ from the message, and unicasts the following message to each neighbor except the node who sent the request to J earlier (here, this is S):

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

where $\text{MAC}_{J,N_x^J}^{\text{prv}}$ 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. 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 receives that.

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

$$D \rightarrow Z : (\text{RREP}, \text{rnd}, \text{addr}_D, \text{label}_{Z \rightarrow S}^{\text{Out}}, \text{label}_{D \rightarrow D}^{\text{In}}, \text{MAC}_{D,S}, \text{MAC}_{D,Z}^{prv})$$

where rnd is the request id received in the corresponding route request message, $MAC_{D,S}$ is the message authentication code generated by D on the elements of the above message excluding addr_D , $\text{label}_{Z \rightarrow S}^{\text{Out}}$, and $\text{label}_{D \rightarrow D}^{\text{In}}$ using the pairwise key shared with S . Then, D generates previous-hop MAC $MAC_{D,Z}^{prv}$ on all elements of the message. Receiving this unicast message, Z first checks the authenticity of the message by verifying $MAC_{D,Z}^{prv}$. If this is successful, Z replaces $MAC_{D,Z}^{prv}$ with $MAC_{Z,K}^{prv}$, and sends the message directly to node K , from which Z received the corresponding request message identified by rnd :

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

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

10.1.1 Computation and communication overhead

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 CBC-MAC construction with a common block cipher like Skipjack for MAC computation as proposed in [37], a MAC has a size of 64 bits. Therefore, there is 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 that 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 2.x is suggested to be around 36 bytes [1] 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 [84] or local broadcast keys like in LEAP [90]? In the latter case, 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. In the former case, digital signatures incur a substantial computation overhead. Although recent advances in the public key cryptography (PKC) of sensor networks are very promising [29], 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 [19] 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 [37] and consumes about 0.034 mWs [64]. 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 $0.209 \mu\text{Ws}$ per bit for the transmission and $0.226 \mu\text{Ws}$ per bit for the reception. Thus, as the size of a request packet is 33 bytes (including the header of the packet) under TinyOS 2.x [1], 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 [84] [64] implementation with the shortest key-size (i.e., 160 bits) the signature generation and verification consumes 26.96 mWs and 53.42 mWs [64], 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 [74], 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 [64] which would also incur substantial network delay.

10.2 Security analysis

In this Subsection, we prove that Secure-TinyLUNAR is indeed secure in our simulation-based model adapted from Section 5 to secure label-switching routing.

10.2.1 The model

Static model: The static model is described in 5.2. According to the definition of the *cost function* in Subsection 5.2, $\mathcal{C} : V \rightarrow \mathbb{R}$ assigns the minimal delay of routing messages to each node in the network (i.e., the minimal delay that the particular node can cause in the travel of the message). We assume that $\mathcal{C}(v^*) = 0$ for all $v^* \in V^*$.

Security objective function: Before introducing the security objective function [96] of secure label-switching routing, we introduce some definitions in order to ease its formalization.

Definition 2 (Anchor entry) An anchor entry $(v_{src}, v_{dest}, \text{addr}_{next}, \text{label}_{v_{src} \rightarrow v_{dest}}^{\text{Out}}, \text{delay}_{v_{src}, v_{dest}})$ 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_{next} , the outgoing label of the source towards the destination is $\text{label}_{v_{src} \rightarrow v_{dest}}^{\text{Out}}$, and the delay of the quickest path through addr_{next} to the destination is $\text{delay}_{v_{src}, v_{dest}}$.

Definition 3 (Intermediate entry) An intermediate entry $(v_{im}, \text{addr}_{next}, \text{label}_{v_{im} \rightarrow v_{dest}}^{\text{In}}, \text{label}_{v_{im} \rightarrow v_{dest}}^{\text{Out}})$ is the representation of a routing entry at an intermediate node v_{im} , where the next-hop towards the destination has (local) address addr_{next} , the incoming label and the outgoing label of v_{im} towards the destination are $\text{label}_{v_{im} \rightarrow v_{dest}}^{\text{In}}$ and $\text{label}_{v_{im} \rightarrow v_{dest}}^{\text{Out}}$, respectively.

Definition 4 (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 .

Definition 5 (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 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 $E_{i,j} = 1$.

Definition 6 (Workable path) A sequence of nodes $(v_{\ell_0}, v_{\ell_1}, \dots, v_{\ell_{d-1}}, v_{\ell_d})$ 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 ($v_{\ell_i}, v_{\ell_{i+1}} \in V \setminus V^*$).

The state of the system is represented by the ensemble of all anchor and intermediate entries of all nodes.

Definition 7 (Correct state) A state is correct with respect to configuration $conf$, if for every anchor entry $r_0 = (v_{src}, v_{dest}, \text{addr}_{next}, \text{label}_{v_{src} \rightarrow v_{dest}}^{\text{Out}}, \text{delay}_{v_{src}, v_{dest}})$, where $v_{src}, v_{dest} \in V \setminus V^*$, there exists a sequence of intermediate entries $r_i = (v_{\ell_i}, \text{addr}_{next}, \text{label}_{v_{\ell_i} \rightarrow v_{dest}}^{\text{In}}, \text{label}_{v_{\ell_i} \rightarrow v_{dest}}^{\text{Out}})$ ($1 \leq i \leq d$) of honest nodes such that

- $v_{\ell_d} = v_{dest}$ and $\text{label}_{v_{dest} \rightarrow v_{dest}}^{\text{Out}}$ is an application identifier of v_{dest} ,

- $(v_{src}, v_{\ell_1}, \dots, v_{\ell_{d-1}}, v_{dest})$ is a workable path, where $v_{src} = v_{\ell_0}$
- if $v_{\ell_{i-1}}$ and v_{ℓ_i} are direct but not pseudo neighbors then r_{i-1} matches r_i ,
- $\sum_{j=1}^{d-1} \mathcal{C}(v_{\ell_j}) \leq \text{delay}_{v_{src}, v_{dest}}$ (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})

The security objective function $\mathcal{F} : \mathbb{G} \times \mathbb{S} \rightarrow \{0, 1\}$ of secure label-switching routing 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.

Dynamic model: The dynamic model is detailed in Subsection 5.4. In addition to this model, we assume that during a simulation run the maximum lifetime of each entry is set to ∞ . The security objective function is applied to the output of this model (i.e., the ensemble of all routing entries which is the system state itself) in order to decide whether the protocol functions correctly or not.

Definition of secure label-switching routing: We denote the security parameter of the model by κ , which is the key length of the employed MAC scheme in the routing protocol.

Definition 8 A label-switching routing protocol is secure, if for any configuration *conf* and any adversary \mathcal{A} , the probability that $\text{Out}_{conf, \mathcal{A}}^{\mathcal{F}}$ equals to zero is a negligible function of κ .⁶

More intuitively, if a secure routing protocol is secure regarding \mathcal{F} , then any system using this routing protocol may not satisfy the security objective represented by \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 .

10.2.2 Tolerable imperfections of the model

Before proving the security of Secure-tinyLUNAR, we explain the tolerable imperfections of our model in more details. 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 our 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 definitions of workable path and pseudo neighbors in Definition 7 is that two adversarial nodes, who may be located on different network parts, are able to transfer the MACs of honest nodes by either using out-of-band channels like wormholes, or some in-band channels (assuming that these nodes believe that they are neighbors and share the corresponding keys). In the latter case, MACs are transferred as a part of an existing message to remote adversarial nodes. For instance, one adversarial node captures the MAC of an honest neighbor denoted by H , then fragments the MAC, and puts these fragments into new RREQ messages as their random identifiers destined to a remote adversarial node. When this remote adversarial node receives all fragments, it can successfully impersonate H by reconstructing the MAC from the fragments. As these RREQ messages are originated from an adversarial node who may have compromised keys, they will pass all verifications done by intermediate nodes. 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 [11].

The reason that we 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 MAC becomes obsolete. Second, these attacks can be mitigated but, to the best of our knowledge, they are not avoidable completely. Therefore, we consider these attacks as some of the tolerable imperfections of our model.

⁶A 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 $x > N_c$), $\mu(x) \leq x^{-c}$

The third point in Definition 7 requires that direct but not pseudo neighbors on the route must have a matching entry. 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 we cannot make any restrictions on the corresponding entries of $v_{\ell_{i-1}}$ and v_{ℓ_i} , as the adversary can modify the message received from v_{ℓ_i} at her own wish before sending that to $v_{\ell_{i-1}}$. Thus, we rather tolerate this kind of mismatching. Now, let us assume that $v_{\ell_{i-1}}$ and v_{ℓ_i} are 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 $v_{\ell_{i-1}}$. Hence, we also tolerate this kind of mismatching in our model.

Finally, the last point in Definition 7, 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 her. In this way, she can make the cost of each route appear to be higher than it really is that we tolerate 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.

10.2.3 Proof of Security

Theorem 2 *Secure-TinyLUNAR is a secure label-switching routing protocol, if the MAC scheme is secure against existential forgery.*

Proof (sketch) We 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, we 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 $r_0 = (v_{src}, v_{dest}, \text{addr}_{next}, \text{label}_{v_{src} \rightarrow v_{dest}}^{\text{Out}}, \text{delay}_{v_{src}, v_{dest}})$, but there does not exist a workable path between v_{src} and v_{dest} with $\text{label}_{v_{src} \rightarrow v_{dest}}^{\text{Out}}$ as an application identifier.
- Case 2: There exists an anchor entry $r_0 = (v_{src}, v_{dest}, \text{addr}_{next}, \text{label}_{v_{src} \rightarrow v_{dest}}^{\text{Out}}, \text{delay}_{v_{src}, v_{dest}})$ and there exists a workable path $(v_{src}, v_{\ell_1}, \dots, v_{\ell_{d-1}}, v_{dest})$ between v_{src} and v_{dest} , but there does not exist a sequence of intermediate entries $r_i = (v_{\ell_i}, \text{addr}_{next}, \text{label}_{v_{\ell_i} \rightarrow v_{dest}}^{\text{In}}, \text{label}_{v_{\ell_i} \rightarrow v_{dest}}^{\text{Out}})$ ($1 \leq i \leq d$) such that r_{i-1} does match r_i if $v_{\ell_{i-1}}, v_{\ell_i}$ are direct but not pseudo neighbors for all i .
- Case 3: There exists an anchor entry $r_0 = (v_{src}, v_{dest}, \text{addr}_{next}, \text{label}_{v_{src} \rightarrow v_{dest}}^{\text{Out}}, \text{delay}_{v_{src}, v_{dest}})$ and there exists a sequence of intermediate entries $r_i = (v_{\ell_i}, \text{addr}_{next}, \text{label}_{v_{\ell_i} \rightarrow v_{dest}}^{\text{In}}, \text{label}_{v_{\ell_i} \rightarrow v_{dest}}^{\text{Out}})$ ($1 \leq i \leq d$) where $(v_{src}, v_{\ell_1}, \dots, v_{\ell_{d-1}}, v_{dest})$ is a workable path and r_{i-1} matches r_i if $v_{\ell_{i-1}}, v_{\ell_i}$ are direct but not pseudo neighbors for all i , but $\sum_{j=1}^{d-1} \mathcal{C}(v_{\ell_j}) > \text{delay}_{v_{src}, v_{dest}}$.

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 $\text{MAC}_{v_{dest}, v_{src}}$. Let us assume that the adversary \mathcal{A} cannot forge $\text{MAC}_{v_{dest}, v_{src}}$. Thus, $\text{MAC}_{v_{dest}, v_{src}}$ 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 $\text{label}_{v_{dest} \rightarrow v_{dest}}^{\text{Out}}$ is an application identifier. Moreover, as $\text{MAC}_{v_{dest}, v_{src}}$ is received by v_{src} , there exists a sequence of nodes $(v_{s_0}, v_{s_1}, \dots, v_{s_{r-1}}, v_{dest})$ such that $v_{s_{i-1}}, v_{s_i}$ are direct or pseudo neighbors for all $1 \leq i \leq r$, where $v_{src} = v_{s_0}$ and $v_{dest} = v_{s_r}$. This means that there is a workable path between v_{src} and v_{dest} which is a contradiction.

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 $(v_{\ell_0}, \dots, v_{\ell_d})$ between v_{src} and v_{dest} , there is at least one pair $v_{\ell_{i-1}}, v_{\ell_i}$ of honest nodes which are direct and not pseudo neighbors but have no matching entries in their tables. 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 $MAC_{v_{dest}, v_{src}}$. Thus, based on Case 1, there exists a workable path $v_{\ell_0}, \dots, v_{\ell_d}$ 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, 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 $MAC_{v_{\ell_i}, v_{\ell_{i-1}}}^{prv}$ can only be generated by v_{ℓ_i} , $v_{\ell_{i-1}}$ received an msg' message ($msg' \neq msg$) with previous-hop MAC $MAC_{v_x, v_{\ell_{i-1}}}^{prv}$, where $MAC_{v_x, v_{\ell_{i-1}}}^{prv} \neq MAC_{v_{\ell_i}, v_{\ell_{i-1}}}^{prv}$. Since $MAC_{v_{dest}, v_{src}}$ travelled through workable path $(v_{\ell_0}, \dots, v_{\ell_d})$, v_x is an adversarial node and the adversary obtained $MAC_{v_{dest}, v_{src}}$ 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. 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, $delay_{v_{src}, v_{dest}}$ denotes the delay of the travel of $MAC_{v_{dest}, v_{src}}$ from its originator to v_{src} (either as a part of a RREQ or a RREP control message). Let us assume that $MAC_{v_{dest}, v_{src}}$ cannot be forged by the adversary \mathcal{A} . Thus, based on Case 1 and Case 2, $MAC_{v_{dest}, v_{src}}$ is received on workable path $(v_{\ell_0}, \dots, v_{\ell_d})$. As the node costs represent the minimum message delays at the nodes and the adversary cannot reduce the delay at the non-corrupted nodes, $\sum_{j=1}^{d-1} \mathcal{C}(v_{\ell_j}) \leq delay_{v_{src}, v_{dest}}$, which is a contradiction. 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. ■

10.3 Implementation and performance evaluation

In this subsection, we compare the performance of TinyLUNAR and Secure-TinyLUNAR. Based on the available nesC source code of TinyLUNAR under TinyOS 2.x, we implemented Secure-TinyLUNAR by extending TinyLUNAR with the security mechanisms described in the previous subsection⁷. In this way, Secure-TinyLUNAR and TinyLUNAR are not separated chunks of code inside TinyOS, but they are integrated together (they reside in the same source code files), and the application designer can decide which one he wishes to use by defining the `SECURE_TINYLUNAR` macro before compilation. In the lack of this macro, TinyLUNAR is compiled without security extensions. Therefore, the interface description of Secure as well as non-Secure-TinyLUNAR are identical. Regarding the security mechanisms, we employed CBC-MAC using Skipjack as the block-cipher algorithm. They are available as parts of package TinySec [37] that is the standard security package of TinyOS 2.x.

For performance evaluation, we used TOSSIM [46] which is a packet-level simulator for TinyOS 2.x. For energy measurement, we extended TOSSIM with Powertossim 2 [72] that can be downloaded from the contrib part of the TinyOS 2.x distribution. Although there exist different sensor network simulators, either they are not capable of simulating several hundreds of nodes, or they require the application to be described in a different language than nesC. In order to measure network delay, we extended the debug capability of TOSSIM. In particular, each debug message is time stamped, where time is measured according to the simulation time line of TOSSIM. A debug message is always printed out along with its timestamp.

10.3.1 Module description

As TinyLUNAR, Secure-TinyLUNAR directly interfaces to upper layers protocols or applications that need a routing functionality in order to reactively establish a multihop path towards parametrically defined destination.

The label switching forwarder is used in conjunction with Secure-TinyLUNAR routing logic and other control protocols that use label switching forwarding functionality.

The structure of the Secure-TinyLUNAR component is shown in Figure 9. TinyLUNAR provides standard Send interface to upper layer applications that require routing functionality. The implementation of this component contains formation of control routing messages, including the necessary MAC computations, and

⁷The code is available in the UbiSec&Sens subversion repository.

handlers of route request and route reply messages respectively. The component interfaces with SocketC component which provides implementation of link layer label switching forwarding. The security mechanisms in Secure-TinyLUNAR resides in TinyLUNARC exclusively, thus we do not detail SocketC further.

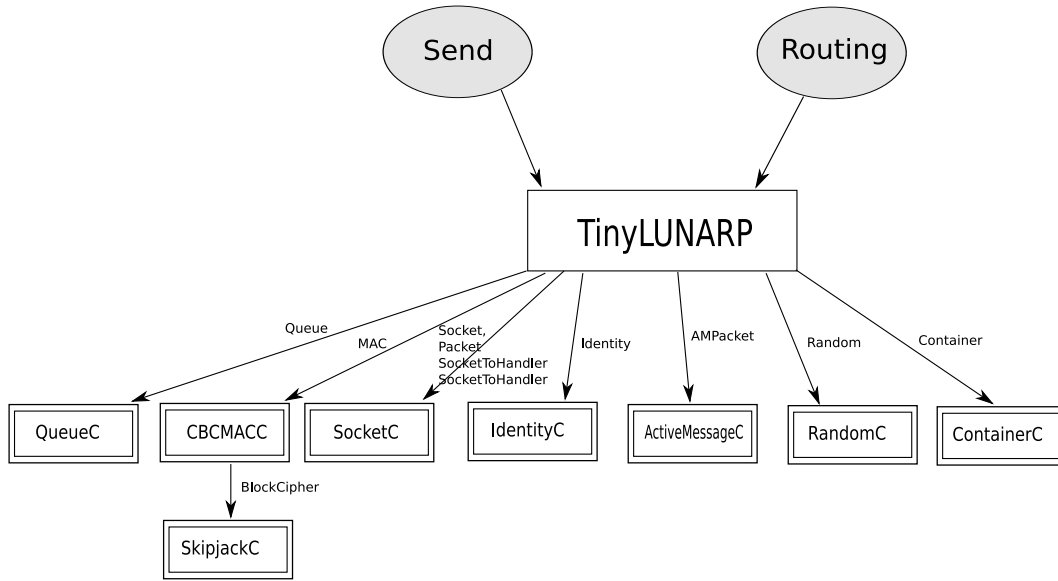


Figure 9: Structure of the (Secure) TinyLUNARC component.

10.3.2 Simulation environment

Powertossim supports only MICA2 [18] motes with radio transceiver Chipcon CC1000 [17] and the ATmega128L [16] micro controller. We configured all nodes to operate at 900 MHz (output power set to 0 Dbm and the transmission data rate is 19.2 Kbps) and the receive sensitivity is set to -110 dBm for all topologies.

Node deployment: We perform simulations on network topologies both with symmetric and asymmetric links. The reason is that neither TinyLUNAR nor Secure-TinyLUNAR incorporates any asymmetric link detection mechanism. Furthermore, we investigate 4 network topologies with 50, 200, 500, and 700 nodes with rectangular network fields of sizes 250×250 , 600×600 , 1000×1000 , and 1200×1200 metres, resp. Nodes are deployed uniformly at random. Considering these topologies and the above settings of the simulator, the average number of neighbors is around 12 for all topologies.

Radio characteristics: According to the requirements of the UbiSec&Sens agriculture and vehicular scenarios, we assume an outdoor environment with near ground transmissions. Thus, we approximate the environmental parameters according to [75] and [91]. The link gains between nodes are calculated by following the log-normal path-loss model. In this model, the received power at distance d is calculated by formula

$$P_r(d) = P_t - PL(d_0) - 10\eta \log_{10} \left(\frac{d}{d_0} \right) + \mathcal{N}(0, \delta)$$

Here, P_t denotes the transmission (output) power, $PL(d_0)$ is the path-loss at reference distance d_0 , η is the path-loss exponent, and $\mathcal{N}(0, \delta)$ is a Gaussian random variable with mean 0 and variance δ (standard deviation due to multipath effects).

In our simulation model, a packet is delivered from a sender s to receiver r , if the following conditions are satisfied:

1. $P_r(d) - N < SNR_{threshold}$, where d is the distance between node r and node s , N is the environmental noise, and $SNR_{threshold}$ is the predefined SNR (Signal-to-Noise Ratio) value adjusted by according to

the PRR (packet reception rate) – SNR characteristic curve of CC1000 [75]. In all simulation runs, we set $SNR_{threshold}$ to 10.2 dBm which corresponds to 0.9 PRR.

2. $P_r(d) > S$, where S denotes the receive sensitivity of the radio chip,
3. the radio of the receiver is turned on and it is in Rx mode,
4. and the receiver does not receive any other packet simultaneously.

If any of the above conditions does not hold, the packet is dropped.

MAC layer: PowerTossim uses the standard TinyOS 2 CC1000 radio stack, which includes the B-MAC detailed in [65]. B-MAC is a CSMA protocol that employs clear channel assessment (CCA) and packet backoffs for channel arbitration, link layer acknowledgements for reliability, and low power listening (LPL) for low power communication. In our simulation runs, we use the default B-MAC settings (i.e., CCA is enabled, LPL is disabled, the initial back-off time and the back-off time in case of congestion are chosen uniformly at random).

Simulation scenario: For each simulation run on each topology, we select a source-destination pair uniformly at random, where the source initiates a topology discovery towards the destination. After a successful discovery, the source sends a data packet containing 6 bytes. The simulation ends when the destination receives the data packet, or the maximum simulation time expires. This simulation run is repeated for both TinyLUNAR and Secure-TinyLUNAR in order to provide sufficient number of patterns for statistical measurements. Although a source-destination pair is selected randomly for each run, we provide the pseudo random number generator used in TOSSIM with predefined seed, whereby we achieve that Secure-TinyLUNAR and TinyLUNAR are executed with the same source-destination pair in each run.

10.3.3 Figures of merit

Depending on the routing objectives, there is a multitude of utility functions of routing protocols in wireless sensor networks. As TinyLUNAR uses time as the default routing metric, we measure network delay during the simulation runs. Particularly, we measure the time which elapses between the generation of the RREQ message at the source node and the reception of the data packet at the destination. This can be further divided into two parts: the delay of the route discovery process (from the generation of the RREQ until the reception of the respective RREP message at the source node) and the delay of data forwarding (from the transmission of the data packet at the source until its reception at the destination).

In addition, all routing protocols put a great effort on minimizing energy consumption (including TinyLUNAR), thus we also measure the average of all nodes' energy consumption. In particular, we evaluate the computational and transmission costs by measuring the energy consumed by the Micro Controller Unit (MCU) and by the radio transceiver, resp.

Finally, we also evaluate the delivery ratio in those scenarios where links can be asymmetric. For the symmetric case, all packets were delivered successfully as radio interference is not considered in Powertossim 2.

10.3.4 Simulation results

Delivery ratio: The delivery ratio is shown in Table 1.

The most common reason for the degraded packet delivery ratio is that RREP messages cannot be delivered back to the source from the destination, and thus the corresponding (pending) routing entries caused by the RREQ messages are purged after a certain time. This frequently happens in networks having nodes with asymmetric links, as the reception of an RREQ message does not imply that the sender is reachable. Although there exist several mechanisms to detect and eliminate asymmetric links from route discovery process, TinyLUNAR, and thus Secure-TinyLUNAR, do not use either of them. That is the reason that we decided to evaluate the performance of both protocols in case of symmetric links too.

<i>Topology</i>	<i>Secure-TinyLUNAR</i>	<i>TinyLUNAR</i>
50 nodes	0.628	0.758
200 nodes	0.4401	0.4832
500 nodes	0.2436	0.3034
700 nodes	0.2159	0.25

Table 1: Resulted delivery ratio with different network sizes. Links between nodes can be asymmetric.

The delivery ratio of Secure-TinyLUNAR is slightly worse than the delivery ratio of TinyLUNAR, as the probability of discovering a longer route towards the destination in terms of hop count is greater, and thus asymmetric links also occur more likely. Here we note that a node which receives a previous-hop MAC processes the RREQ message immediately and forwards that in our implementation. Thus, it might occur that neighbors that are farther from the destination in terms of hop count has a bigger chance to deliver the RREQ message to the destination sooner than those nodes which are closer but receive the corresponding previous-hop later. This results in better network delay due to the less channel contention but causes worse delivery ratio.

The reason that delivery ratio degrades with network sizes is straightforward. In larger networks, there are more intermediate nodes between a randomly selected node pair on average than in smaller networks, and thus the probability that there is at least one asymmetric link on the discovered route is also higher.

Energy consumption: The energy consumption in case of symmetric links is shown in Figure 10. One can observe that while in small networks the radio and mcu energy consumption is comparable in case of Secure-TinyLUNAR, in large networks the energy spent on radio communication is approx. 30% larger than the energy consumed by the MCU. The reason is that route discovery takes more time in larger networks on average (see below) which results in prolonged idle time for the radio transceiver. As low-power listening is not enabled by default, the transceiver consumes considerably energy even if it is not used. This also causes the overall energy consumption in larger networks to be greater than in smaller networks.

It is also easy to see that the computation overhead of TinyLUNAR and Secure-TinyLUNAR is almost the same that might be surprising at first glance (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 in small networks and 9 times larger in large networks than in TinyLUNAR. 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. Note that the communicational overhead is slightly larger in Secure-TinyLUNAR in small networks in case of asymmetric links than in case of symmetric links. However, in general, as one can see in Figure 11, energy consumption does not vary in case of asymmetric links compared to the symmetric case because only some RREP messages and the corresponding data messages are failed to be delivered which do not influence the energy consumption substantially (they are propagated along with a single path, whereas route requests flood the entire network).

Network delay: The network delay is shown in Figure 10 for symmetric links, and in Figure 11 for asymmetric links. In symmetric case, the network delay of Secure-TinyLUNAR is about 7 times larger than the network delay of TinyLUNAR in all networks. The reason is that the usage of the "pseudo broadcast" scheme in Secure-TinyLUNAR can cause substantial network delay. In particular, a node can only forward a RREQ message if the corresponding previous-hop MAC is received. However, the sender needs to transmit a MAC per neighbor which may take considerable time due to channel contention. One can observe that about 85% of the total delay is the route discovery delay and only 15% is the data forwarding delay.

In asymmetric case, the network delay are slightly better in all networks than in symmetric case. Particularly, the total network delay decreases with about 12%. This can be explained by the less number of transmissions caused by asymmetric links.

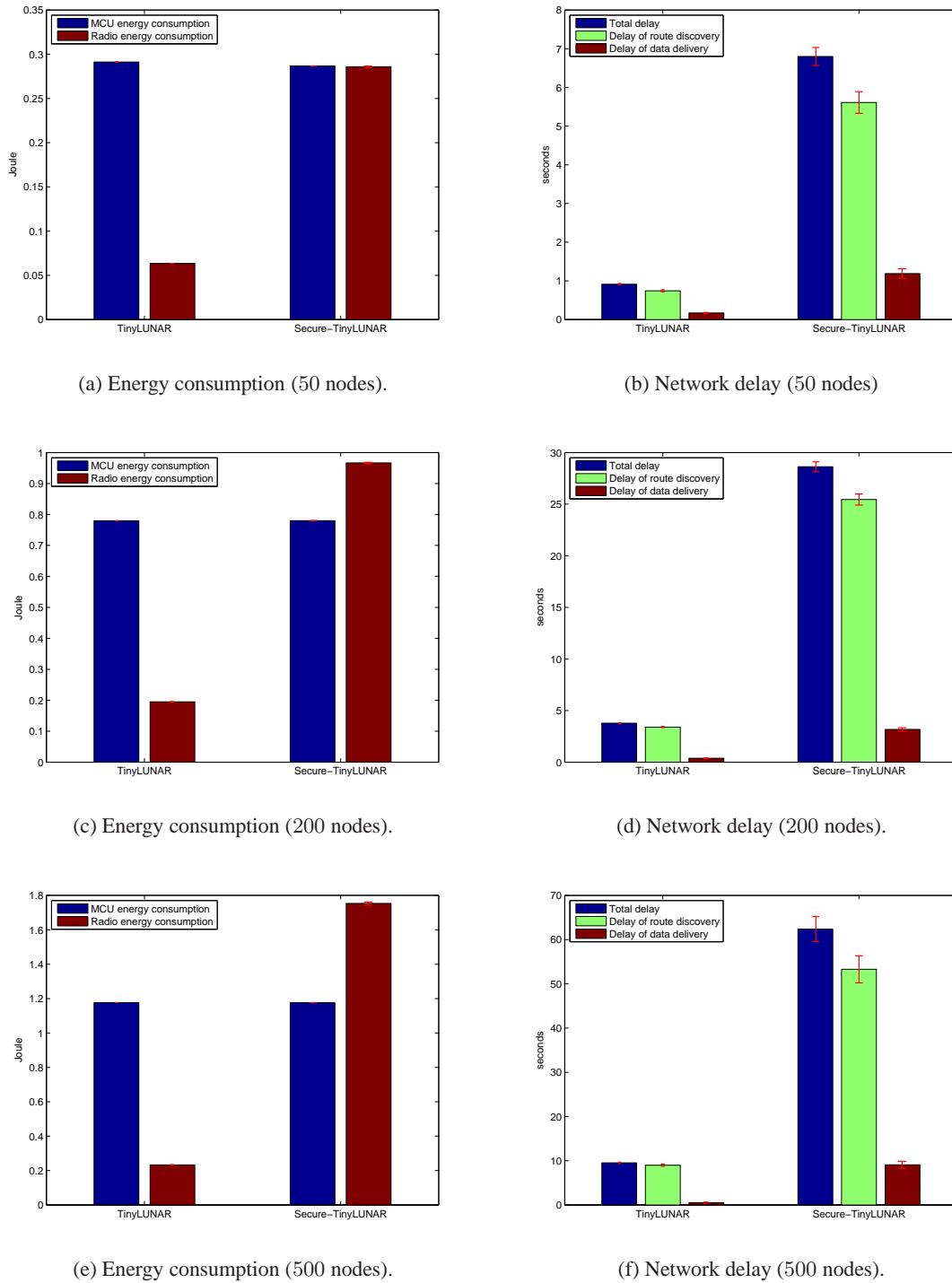
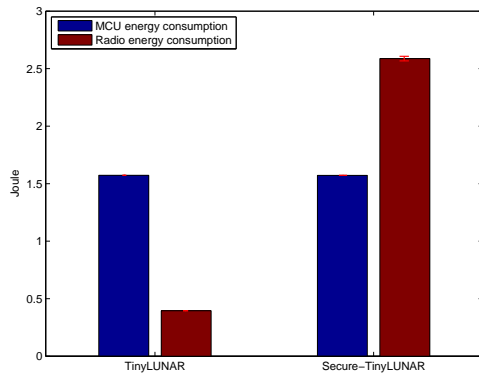


Figure 10: Energy consumption and network delay measurements in case of symmetric links. The whiskers on the top of the bars correspond to the 95% confidence interval of the average.

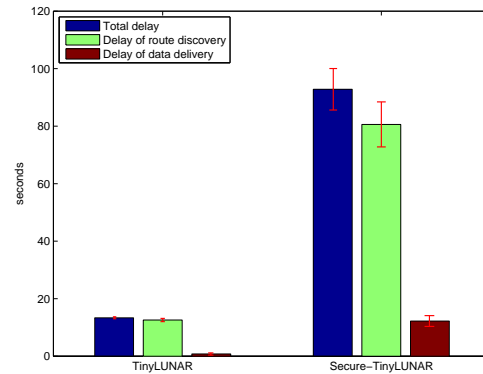
11 Summary

We developed a secure label-switching routing protocol for wireless sensor networks, called Secure-TinyLUNAR. Secure-TinyLUNAR is the secure variant of TinyLUNAR, which is an efficient reactive routing protocol for wireless sensor networks. After showing that TinyLUNAR is vulnerable to several impersonation attacks, we designed Secure-TinyLUNAR, which 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



(g) Energy consumption (700 nodes).



(h) Network delay (700 nodes).

Figure 10: Energy consumption and network delay measurements in case of symmetric links. The whiskers on the top of the bars correspond to the 95% confidence interval of the average.

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 need not verify this MAC saving some resources. Nevertheless, the protocol is provably secure, even if these MACs are not verified by intermediate nodes.

We adapted our simulation-based model to secure label-switching routing, and showed that Secure-TinyLUNAR is provably secure in this model. This model is only concerned with attacks aiming to corrupt the routing entries, different attacks like DoS attacks or rushing are not considered. For instance, Secure-TinyLUNAR is exposed to DoS attacks where unauthentic forged control messages can traverse several hops before being dropped.

Finally, we implemented and evaluated the performance of Secure-TinyLUNAR in TOSSIM under TinyOS 2.x.⁸ We measured the network delay, energy consumption and delivery ratio in different networks containing symmetric and asymmetric links. We concluded that while the consumed computational energy is comparable to TinyLUNAR, the employed security mechanisms introduce substantial communicational costs and network delay. This is mainly caused by the usage of unicast communications instead of broadcast communications between nodes in the route discovery phase. However, we analytically justified that this is still more favourable than employing digital signatures with broadcast communication.

12 Related work

In [38], 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 [5, 63], 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 [38], but the capabilities of an adversarial node regarding message manipulations are not discussed.

The simulation paradigm is described in [63, 14]. These models were mainly proposed with wired networks in mind typically implemented on the well-known Internet architecture, and the wireless context is not

⁸The code is available in the UbiSec&Sens subversion repository.

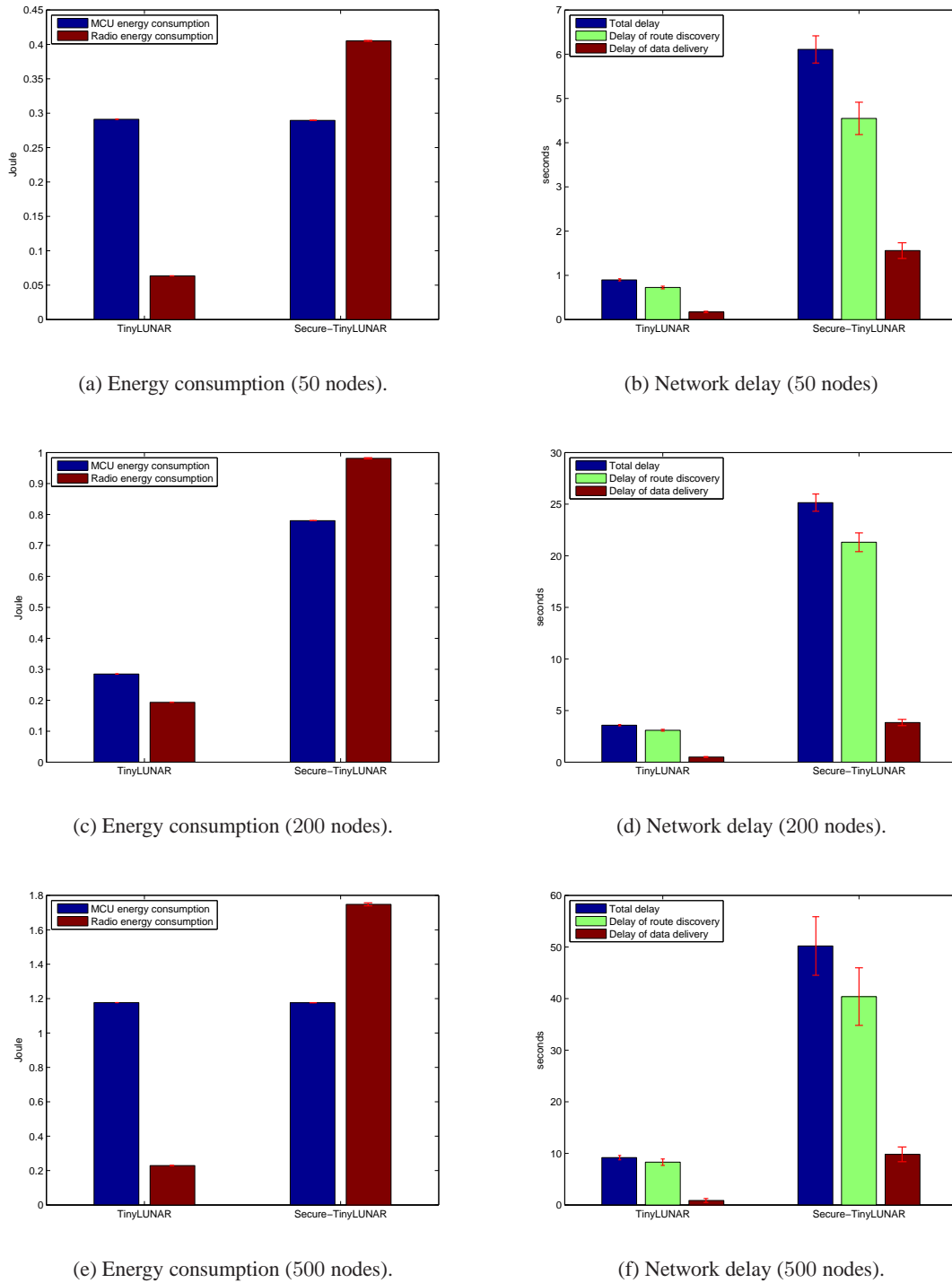
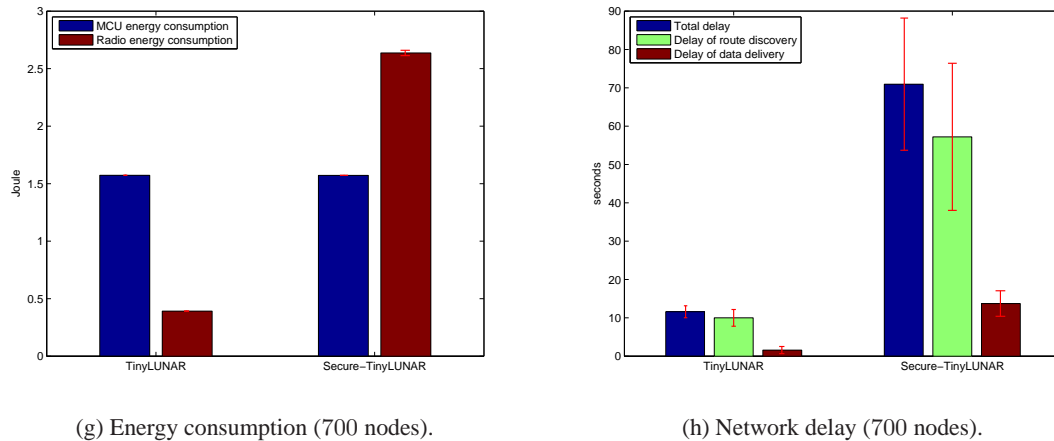


Figure 11: Energy consumption and network delay measurements in case of asymmetric links. The whiskers on the top of the bars correspond to the 95% confidence interval of the average.

focused there. In our opinion, the multi-hop nature of communications is an inherent characteristic of wireless sensor networks, therefore, it should be explicitly modelled. In more particular, the broadcast nature of communication enables a party to overhear the transmission of a message that was not destined to him, however, this transmission can be received only in a certain range of the sender. The size of this range is determined by the power at which the sender sent the message. Another deviation from [63] is the usage of the security objective function in the definition of security. In [63], the indistinguishability is defined on the view of the honest parties (on their input, states, and output) in the ideal-world and in the real-world models. However, an adversary can distort the states of the honest parties in unavoidable ways, and hence, the classical definition



(g) Energy consumption (700 nodes).

(h) Network delay (700 nodes).

Figure 11: Energy consumption and network delay measurements in case of asymmetric links. The whiskers on the top of the bars correspond to the 95% confidence interval of the average.

would be too strong and no routing protocol would satisfy it. On the other hand, our model is compliant with [63] considering high-level connections between nodes. In [63], the standard cryptographic system allows us to define each high-level connection as secure (private and authentic), authenticated (only authentic), and insecure (neither private nor authentic). In this taxonomy, the communication channel between two honest nodes can be either insecure or secure in our model. If an adversarial node is placed in the communication range of one of the communicating nodes, then it is considered to be an insecure channel. If the adversary can reach none of the communicating nodes, the channel between that nodes is hidden from the adversary, and thus, it is considered to be secure.

In the literature, there are some prior works [43, 86, 52, 60] that also used formal techniques to model the security of multi-hop routing protocols. However, they were mainly proposed for ad hoc network routing, and they either inherently differ from simulation-based models [86, 52, 60] (the model proposed in [52] is based on CPAL-ES, and the model in [86] is similar to the strand spaces model), or they are limited to model some protocol specific attacks (like rushing) [43].

In contrast to this, in our work, we are concerned with more general security objectives.

Our work is primarily based on [13, 95]. Here, the authors also use the simulation paradigm to prove the security of routing protocols in wireless ad-hoc networks. However, our model differs from the models in [13, 95] in two ways:

- *Adversary model*: The adversary in [13] and [95] is assumed to have the same resources and communication capabilities as an ordinary node in the network. Therefore, that adversary model deviates from the so-called Dolev-Yao model in [21]. In our work, the adversary also uses wireless devices to attack the systems, and it is reasonable to assume that the adversary can interfere with communications only within its power range. The adversarial nodes belonging to the sensor-class nodes has the same resources and communication capabilities as an ordinary sensor node, but a more resourced adversarial node (e.g., laptops) may affect the overall communication of an entire part of the network depending on the power range of the resourced adversarial device. That resourced devices also make the adversary able to perform more sophisticated message manipulations.
- *Modelling security objectives*: In ad hoc networks, nodes construct routes between a source and a destination [61, 36], whereas sensor nodes should build a complete routing topology for the entire network. In case of sensor networks, the only destination for all nodes is the base station [3]. In addition, sensor nodes are resource constrained, which implies that we also need to model the energy consumption of sensor nodes, since several attacks impacts the network lifetime. These differences from ad hoc networks has yielded a wide range of sensor applications, and thus, sensor routing protocols [3] are much diverse than ad hoc routing protocols. Hence, the security objectives cannot be modelled uniformly for sensor routing protocols.

TinyLUNAR is a reactive routing protocol for wireless sensor networks that is proposed in [59]. The main design objective of TinyLUNAR was to support multiple communication patterns for both data-centric and address-centric communications, where functional universality is gained at the expense of increased complexity. TinyLUNAR is based on LUNAR (Lightweight Underlay Ad hoc Routing) which is an ad hoc network routing protocol [78] employing the label switching (or virtual circuit) routing paradigm. By adopting this paradigm, the authors showed that it is feasible to implement TinyLUNAR under TinyOS 2.x using only one byte field of the IEEE 802.15.4 MAC header [26] per packet for making packet forwarding decisions during the data forwarding phase. This makes TinyLUNAR a more effective routing protocol than e.g., tinyAODV (which is the adaptation of AODV [61] to wireless sensor networks) in such networks where nodes are stationary or show moderate mobility.

Many secure routing protocols have been proposed for wireless ad hoc networks [32], from which ARAN [68] is the most related to Secure-TinyLUNAR. Similar to ARAN, Secure-TinyLUNAR also uses time as the default routing metric. However, compared to ARAN that uses two signatures per routing messages, Secure-TinyLUNAR considers the specifics of sensor nodes and employs only MACs, where an intermediate node performs only two MAC generations. In addition, MACs have a shorter size and less computation overhead than signatures. Besides that, Secure-TinyLUNAR is also provably secure in a similar simulation-based model like ARAN.

There have been proposed some secure routing protocols for wireless sensor networks [83] [20]. In [20], the authors propose an intrusion tolerant routing protocol, called INSENS, for wireless sensor networks. INSENS is a centralized link-state routing protocol, where each node sends its local neighborhood information to the base station, which then computes the forwarding table of each node. Similar to Secure-TinyLUNAR, INSENS is also provably secure in a simulation-based model adapted to secure link-state routing. However, INSENS is not scalable to large-scale networks due to its centralized nature, and the base station is a single point of failure.

In [83], a family of configurable and secure routing protocols is proposed for wireless sensor networks called SIGF. The authors did not provide a formal security analysis of SIGF, but they evaluated the performance of SIGF in various environments containing malicious nodes. As SIGF consists of position based routing protocols, it is intended for those sensor networks where each node is capable to obtain its geographic position. This assumption holds only for a few sensor applications due to its high induced cost in terms of additional hardware needed in the sensor nodes.

References

- [1] TinyOS 2.x. <http://www.tinyos.net/tinyos-2.x/doc/>, 2007.
- [2] G. Ács, L. Buttyán, and I. Vajda. Modelling adversaries and security objectives for routing protocols in wireless sensor networks. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, 2006.
- [3] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11:6–28, 2004.
- [4] K. Balakrishnan, J. Deng, and P. K. Varshney. Twoack: preventing selfishness in mobile ad hoc networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2137–2142, 2005.
- [5] 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.
- [6] R. Blom. Non-public key distribution. In *Advances in Cryptology (Crypto)*, pages 231–236, 1982.
- [7] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In *Advances in Cryptology (Crypto)*, pages 471–486, 1992.
- [8] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.
- [9] S. Brands and D. Chaum. Distance-bounding protocols. In *Proceedings of the Workshop on Theory and Application of Cryptographic Techniques*, 1993.
- [10] 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.
- [11] M. Burmester and B. de Medeiros. Towards provable security for route discovery protocols in mobile ad hoc networks, 2007.
- [12] L. Buttyán, L. Dóra, and I. Vajda. Statistical wormhole detection in sensor networks. In *Proceedings of the European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS)*, 2005.
- [13] L. Buttyán and I. Vajda. Towards provable security for ad hoc routing protocols. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, 2004.
- [14] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, 2001.
- [15] D. Coppersmith and M. Jakobsson. Almost optimal hash sequence traversal. In *Proceedings of Financial Cryptography (FC)*, 2002.
- [16] ATMEL Corporation. Atmega128l datasheet. http://www.datasheetcatalog.com/datasheets_pdf/A/T/M/E/ATMEGA128L.shtml.
- [17] Chipcon Corporation. CC1000 datasheet. http://www.chipcon.com/files/CC1000_Data_Sheet_2_2.pdf.
- [18] XBow Corporation. Mica2 datasheet. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf.
- [19] CrossBow Technology Inc. MICAz Mote Platform Datasheet. http://www.xbow.com/Products/Product_p 2005.
- [20] 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.

- [21] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.
- [22] J. R. Douceur. The sybil attack. In *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.
- [23] W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, pages 42–51, 2003.
- [24] S. Dulman, T. Nieberg, J. Wu, and P. Havinga. Trade-off between traffic overhead and reliability in multipath routing for wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2003.
- [25] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, pages 41–47, 2002.
- [26] IEEE Standard for Information technology. 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.
- [27] K. Gabriel and R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.
- [28] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly resilient, energy efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review (MC2R)*, 8(2), 2001.
- [29] N. Gura, A. Patel, and A. Wander. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *Proceedings of the 2004 Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004)*, 2004.
- [30] 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.
- [31] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. In *Proceedings of the IEEE Symposium on Network and Distributed Systems Security (NDSS)*, 2004.
- [32] Y.-C. Hu and A. Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy Magazine*, 2(3):28–39, 2004.
- [33] 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.
- [34] Y.-C. Hu, A. Perrig, and D. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *Proceedings of the IEEE Conference on Computer Communications (Infocom)*, 2003.
- [35] 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.
- [36] 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.
- [37] 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.

- [38] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks*, 1, 2003.
- [39] 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.
- [40] 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.
- [41] 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.
- [42] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.
- [43] J. Kong, X. Hong, and M. Gerla. Modeling ad-hoc rushing attack in a negligibility-based security framework. In *Proceedings of the 5th ACM Workshop on Wireless Security (WiSe)*, 2006.
- [44] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, 1981.
- [45] L. Lazos, R. Poovendran, C. Meadows, P. Syverson, and L. Chang. Preventing wormhole attacks on wireless ad hoc networks: a graph theoretic approach. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2005.
- [46] 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.
- [47] Q. Li, J. Aslam, and D. Rus. Hierarchical power-aware routing in sensor networks. In *Proceedings of the DIMACS Workshop on Pervasive Networking*, 2001.
- [48] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, pages 52–61, 2003.
- [49] D. Liu and P. Ning. Multi-level μ TESLA: Broadcast authentication for distributed sensor networks. *ACM Transactions in Embedded Computing Systems (TECS)*, 3(4):800–836, 2004.
- [50] D. Liu, P. Ning, S. Zhu, and S. Jajodia. Practical broadcast authentication in sensor networks. In *Proceedings of the International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous)*, pages 118–129, 2005.
- [51] L. Mark, A. Perrig, and B. Whillock. Seven cardinal properties of sensor network broadcast authentication. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, 2006.
- [52] 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.
- [53] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 2000.
- [54] R. C. Merkle. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1980.
- [55] R. C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology (Crypto)*, 1988.

- [56] V. Miller. Uses of elliptic curves in cryptography. In *Advances in Cryptology (CRYPTO)*, pages 417–462, 1985.
- [57] J. Newsome, R. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004.
- [58] Next-Generation Secure Computing Base (NGSCB). <http://www.microsoft.com/resources/ngscb/de> 2003.
- [59] 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)*, 2007.
- [60] P. Papadimitratosa, Z.J. Haas, and J.-P. Hubaux. How to specify and how to prove correctness of secure routing protocols for manet. In *Proceedings of IEEE CS BroadNets 2006*, 2006.
- [61] 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.
- [62] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. *Wireless Networks Journal (WINE)*, 8, 2002.
- [63] 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.
- [64] 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.
- [65] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *In Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Boston, MA, November 2004.
- [66] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: a geographic hash table for data-centric storage. In *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.
- [67] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *CACM* 21,2, 1978.
- [68] 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.
- [69] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, 2003.
- [70] Y. Sella. On the computation-storage trade-offs of hash chain traversal. In *Proceedings of Financial Cryptography (FC)*, 2003.
- [71] 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.
- [72] 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'04)*, Baltimore, MD, November 2004.
- [73] 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 '98)*, 1998.

- [74] Moteiv Corporation. Tmote sky ultra low power IEEE 802.15.4 compliant wireless sensor module. <http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf>, 2006.
- [75] K. Sohrabi, B. Manriquez, and G. J. Pottie. Near ground wideband channel measurement in 800-1000 mhz. In *Proc. IEEE 49th Vehicular Technology Conference*, Boston, MA, July 1999.
- [76] Trusted Computing Group (TCG). <https://www.trustedcomputinggroup.org/>, 2003.
- [77] G. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268, 1980.
- [78] C. Tschudin, R. Gold, O. Rensfelt, and O. Wibling. LUNAR: a lightweight underlay network ad-hoc routing protocol and implementation. In *Proceedings of the 4th International Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN 2004)*, 2004.
- [79] S. Čapkun, L. Buttyán, and J.-P. Hubaux. SECTOR: Secure tracking of node encounters in multi-hop wireless networks. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, 2003.
- [80] S. Čapkun and J.-P. Hubaux. Secure positioning of wireless devices with application to sensor networks. In *Proceedings of the IEEE Conference on Computer Communications (Infocom)*, 2005.
- [81] S. Čapkun and J.-P. Hubaux. Secure positioning in wireless networks. *IEEE Journal on Selected Areas in Communications*, February 2006.
- [82] W. Wang and B. Bhargava. Visualization of wormholes in sensor networks. In *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, 2004.
- [83] 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.
- [84] ANSI X9.63. The elliptic curve digital signature algorithm (ECDSA), 1999.
- [85] 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.
- [86] 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.
- [87] H. Yih-Chun, M. Jakobsson, and A. Perrig. Efficient constructions for one-way hash chains. In *Proceedings of the Conference of Applied Cryptography and Network Security (ACNS)*, 2005.
- [88] M. G. Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, 2002.
- [89] Q. Zhang, P. Wang, D. S. Reeves, and P. Ning. Defending sybil attacks in sensor networks. In *Proceedings of the International Workshop on Security in Distributed Computing Systems (SDCS)*, pages 185–191, 2005.
- [90] 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.
- [91] Marco Zuniga and Bhaskar Krishnamachari. An analysis of unreliability and asymmetry in low-power wireless links. *ACM Transactions on Sensor Networks*, 3, June 2007.
- [92] G. Ács and L. Buttyán. A taxonomy of routing protocols for wireless sensor networks. *Híradástechnika (English Edition)*, December 2006.

- [93] G. Ács and L. Buttyán. Designing a secure label-switching routing protocol for wireless sensor networks. *submitted to Periodica Polytechnica (<http://www.pp.bme.hu/>)*, June 2008.
- [94] G. Ács and L. Buttyán. Secure routing in wireless sensor networks. 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.
- [95] 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)*, 2005.
- [96] 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)*, 2006.
- [97] 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), 2006.
- [98] 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)*, 2007.
- [99] 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.