Designing a Secure Label-switching Routing Protocol for Wireless Sensor Networks

Gergely Ács, Levente Buttyán
Laboratory of Cryptography and Systems Security (CrySyS)
Department of Telecommunications
Budapest University of Technology and Economics, Hungary
{acs, buttyan}@crysys.hu

December 22, 2008

Abstract

Although a multitude of routing protocols exist for wireless sensor networks developed for various application domains, *secure* sensor network routing protocols do not exhibit such variety. In addition, those few sensor network routing protocols that were developed with security in mind still lack a formal proof of their security properties. In order to remedy this situation, in this paper, we propose a novel secure routing protocol, called Secure-TinyLUNAR, for wireless sensor networks, and we formally prove its security. In our model, security is defined in terms of the correctness of the routing table entries of the honest nodes. Besides its provable security, another advantage of Secure-TinyLUNAR is that, similar to TinyLUNAR, it uses label-switching routing, which results in reduced addressing overhead during data packet forwarding.

1 Introduction

Wireless sensor networks are composed of a large number of resource constrained sensor nodes and possibly a few more powerful nodes called base stations. Sensor nodes usually communicate with their local neighborhood via low-power wireless links, and are capable to sense their immediate surroundings. This sensed data is typically sent to the base station or other sensor nodes in a multi-hop manner by using some of the multitude of routing protocols specifically proposed for these networks [3].

Many applications require sensor networks to operate in hostile environments. Security thus becomes a critical issue for network protocols as well. For instance, by attacking the routing protocol, an adversary can easily degrade the performance of the network; the illegal manipulation of some routing messages can cause shorter network lifetime, degraded packet delivery ratio, or increased network delay[1].

[1] In this work, we consider the security of route (topology) discovery phase of routing protocols.

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 [23] [18] [10]. 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 is the secure variant of TinyLUNAR [14] which is a reactive routing protocol proposed for wireless sensor networks. Using the label-switching routing paradigm, TinyLUNAR has only one byte addressing overhead per packet in the data forwarding phase, which, considering the high communication costs in wireless environment, makes it an efficient routing scheme. Although TinyLUNAR has a slightly greater RAM consumption than other reactive routing protocols like tinyAODV [15], 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 use a formal framework proposed in [27] [29] to analyze the security of Secure-TinyLUNAR. This framework considers those attacks that aim to corrupt the routing entries of the nodes creating incorrect routing state. This also has been successfully used so far to analyze the security of several multi-hop routing protocols like Ariadne [28], endairA [28], SRP [28], ARAN [26], SAODV [26] or INSENS [29]. We further demonstrate the strength of this framework by showing that Secure-TinyLUNAR is also provably secure. In particular, we first adapt this model to secure label-switching routing, and prove that Secure-TinyLUNAR is indeed secure in that model.

The organization of this paper is as follows. In Section 3, we briefly present TinyLUNAR and show some simple impersonation attacks aiming to corrupt the routing tables of honest nodes. Then, in Section 4, we develop Secure-TinyLUNAR which uses MACs to provide defense against the aforementioned attacks. In Subsection 4.2, we first describe the security model of secure label-switching, and then, in Subsection 4.2.3, we prove that Secure-TinyLUNAR is secure in that model. Finally, in Section 5, we conclude our work and discuss future plans.

2 Related work

TinyLUNAR is a reactive routing protocol for wireless sensor networks that is proposed in [14]. 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

[20] 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 [7] 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 [15] 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 [9], from which ARAN [18] 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 [21] [6]. In [6], 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 [21], 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.

In [12], the authors informally analyze the security of some existing sensor network routing protocols. In that paper, routing security is defined implicitly as resistance to some specific attacks, and the proposed countermeasures are only related to these attacks.

Our formal model that we use to analyze the security of Secure-TinyLUNAR is described in [2] [28]. In those papers, a formal model is proposed based on the simulation paradigm [13] to analyze the security of wireless ad hoc and sensor network routing protocols.

3 Security of TinyLUNAR

In this section, we first give a brief overview of the operation of TinyLUNAR. One can read a more detailed description in [14]. Then, we show simple attacks against TinyLUNAR, whereby we motivate the development of Secure-TinyLUNAR.

3.1 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:

$$S \rightarrow * : (\mathsf{RREQ}, \mathsf{rnd}, S, D, addr_S, label^{\mathsf{In}}_{S \rightarrow S}) \qquad \text{(Msg-1)}$$

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

A node J receiving this broadcast message checks whether it has been received the request earlier based on rnd, S, and D. If so, J silently drops the request. Otherwise, J stores the quadruple $(addr_S, label^{\mathsf{In}}_{S \rightarrow S}, \mathsf{rnd}, \mathsf{lifetime})$ in its routing table, where 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:

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

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

$$D \rightarrow Z : (\mathsf{RREP}, \mathsf{rnd}, addr_D, label^{\mathsf{Out}}_{Z \rightarrow S}, label^{\mathsf{In}}_{D \rightarrow D}) \qquad \text{(Msg-2)}$$

where rnd is the random identifier of the corresponding request originated from S, $label^{\mathsf{Out}}_{Z \rightarrow S}$ is the incoming label of Z towards S (i.e., the outgoing label of D towards S) received in the request, and $label^{\mathsf{In}}_{D \rightarrow D}$ is the incoming label of D. Here, $label^{\mathsf{In}}_{D \rightarrow D}$ is a memory address inside the routing table of D and 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 $label^{\mathsf{In}}_{D\rightarrow D}$, 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 $label^{\mathsf{Out}}_{Z\rightarrow S}$ in its memory (routing table), and forwards the message to the node contained by this entry. Let us assume that Z received the corresponding request from node K first. Then, Z sends the following message to K:

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

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

All subsequent nodes receiving the reply do the same operations that Z did. In this way, the *forward traffic flow* is constructed which is defined by the set of all routing entries created at intermediate nodes. This traffic flow is associated with S at the endpoint D. Finally, 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 (Msg-1) 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.

3.2 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 [25] 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 [25].

In the following, we argue that impersonation attacks cause incorrect routing entries in TinyLUNAR. Thus, in Section 4, 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 (Msg-1). For instance, in Figure 1(a), 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 1(b) 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 (Msg-2) 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.

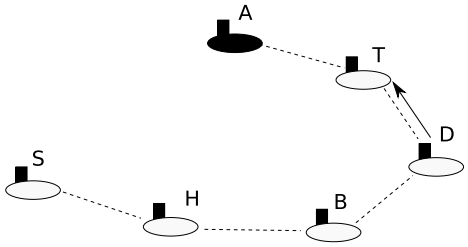
Neighbor impersonation: In Figure 1(c), we illustrate neighbor impersonation attack. The adversarial nodes are A and A'. Assume that H can only be reached by S and A', and the adversary is aware of all nodes' identities and the local addresses of the nodes that she can reach (i.e., local addresses of H, S, B). Furthermore, S wishes to discover a path to D. First, S floods the network with a request (Msg-1) which is received by adversarial node A. A rebroadcasts the request faithfully. However, when the corresponding reply (Msg-2) comes back from D, A rebroadcasts that in the name of H (i.e., A uses H's identity and local network address, which is catched by A'). Finally, receiving this forged reply, S believes that D can be reached through H. However, as H does not receive any replies, it will not forward any messages towards D.

4 Secure-TinyLUNAR

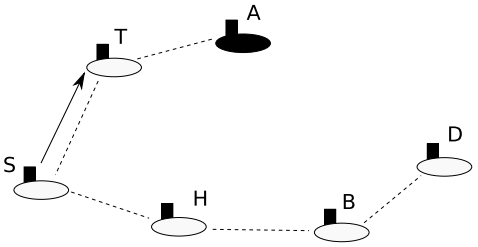
In this section, we first describe the operation of Secure-TinyLUNAR. As a first step, we prevent the impersonation attacks that are described in Subsection 3.2. Secure-TinyLUNAR is the secure variant of TinyLUNAR, where we use pairwise message authentication codes (MACs) to authenticate routing messages between immediate neighbors and also to ensure source/destination authenticity. Finally, in Subsection 4.2.3, we show that this new protocol is provably secure in a model which is adapted to secure label-switching from [29].

4.1 Operation of Secure-TinyLUNAR

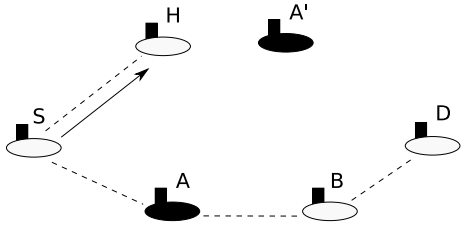
We only discuss the main operational differences with respect to the original (and insecure) TinyLUNAR protocol. We assume that each pair of nodes share a symmetric pairwise key in the network. Any symmetric key pre-distribution schemes proposed for wireless sensor



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



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



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

Figure 1: Impersonation attacks against TinyLUNAR. Dashed lines denote the neighborhood relations, whereas arrows denote the routing entries.

networks (see [30] 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:

$$\text{for all } x,\ S \rightarrow N_x^S : (\mathsf{RREQ}, \mathsf{rnd}, S, D, addr_S, label_{S \rightarrow S}^{\mathsf{ln}}, \mathsf{MAC}_{S,D}, \mathsf{MAC}_{S,N_x^S}^{prv}) \quad \text{(Msg-3)}$$

where $\mathsf{rnd}, S, D, addr_S, label_{S \rightarrow S}^{\mathsf{ln}}$ are the same as in the original TinyLUNAR protocol, $\mathsf{MAC}_{S,D}$ is the message authentication code generated by S on the elements of the message excluding $addr_S$, and $label_{S \rightarrow S}^{\mathsf{ln}}$ using the pairwise key shared with D. After generating $\mathsf{MAC}_{S,D}$, S generates previous-hop MAC $\mathsf{MAC}_{S,N_x^S}^{prv}$ 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 $\mathsf{MAC}_{S,J}^{prv}$. In case it is successful, node J removes $\mathsf{MAC}_{S,J}^{prv}$ 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):

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

where $\mathsf{MAC}^{prv}_{J,N^J_x}$ is the previous-hop MAC generated on all elements of the message using the pairwise key shared between J and N^J_x. 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 $\mathsf{MAC}_{S,D}$ and $\mathsf{MAC}^{prv}_{Z,D}$. If the verifications are successful, D creates the following reply message and sends this directly to node Z:

$$D \rightarrow Z : (\mathsf{RREP}, \mathsf{rnd}, addr_D, label^{\mathsf{Out}}_{Z\rightarrow S}, label^{\mathsf{In}}_{D\rightarrow D}, \mathsf{MAC}_{D,S}, \mathsf{MAC}^{prv}_{D,Z}) \qquad \text{(Msg-4)}$$

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

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

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

4.1.1 Computation and communication overhead

Comparing to TinyLUNAR, Secure-TinyLUNAR requires the sender of a request message (Msg-3) to perform two MAC generations. Furthermore, each node receiving a request (Msg-3) 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 [11], a MAC has a size of 64 bits. Therefore, there is 16 extra bytes in each request (Msg-3) and reply packet (Msg-4). 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 (Msg-3) 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 [22], μTesla [16], or local broadcast keys like in LEAP [24]? In case of local broadcast keys, when a common key is shared among the sender and all its neighbors, cannot guarantee neighbor authentication, as a neighboring adversarial node would be able to impersonate any honest neighbor using the shared key. Although μTesla does not have this drawback and it also uses efficient symmetric cryptography, it requires each receiver to maintain a hash chain [16]. If route discoveries are invoked infrequently, which holds for many sensor networks due to their static nature, the verification of a particular broadcast key requires several evaluations of the employed hash function on average, which can result in significant computational overhead. Moreover, μTesla relies on a clock synchronization protocol which also incurs additional overhead on each node. Finally, digital signatures incur a substantial computation overhead. Although recent advances in the public key cryptography (PKC) of sensor networks are very promising [8], 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 [5] 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 [11] and consumes about 0.034 mWs [17]. If we assume that a node has at most 30 neighbors, all the computation cost is $30 \cdot 0.034 = 1.02$ mWs. If the radio transceiver operates at transmission speed of 250 kbit/s at 3 V supply voltage and the output power is set to 0 dBm (maximum power), then the power consumption is 0.209 μWs per bit for the transmission and 0.226 μWs per bit for the reception. Thus, as the size of a request packet is 33 bytes (including the header of the packet) under TinyOS 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 [22] [17] implementation with the shortest key-size (i.e., 160 bits) the signature generation and verification consumes 26.96 mWs and 53.42 mWs [17], 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 [19], 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 [17] which would also incur substantial network delay.

4.2 Security analysis

In this subsection, we prove that Secure-TinyLUNAR is indeed secure in a simulation-based model adapted from [29] to secure label-switching routing.

4.2.1 The model

Adversary model: We assume that the adversary is represented by adversarial nodes in the network. An adversarial node can be a sensor-class or a laptop-class device. By sensor-class devices, we mean resource constrained devices like ordinary sensor nodes. Laptop-class devices can be more resourced with powerful antennas and unconstrained energy supply like laptops or desktop computers. We further assume that these adversarial nodes can communicate with each other via out-of-band channels (e.g., using other frequency channels or direct wired connections). Moreover, when the adversary captures honest sensor nodes, he may be able to compromise their cryptographic secrets (assuming that such secrets are used in the system). As each adversarial node is assumed to communicate with each other via out-of-band channels, we assume that all adversarial nodes can use all compromised cryptographic secrets.

Generally, the primary goals of the adversary can be degrading the packet delivery ratio, increasing his control over traffic, increasing network delay, and shortening network lifetime depending on the routing objectives. When attacking protocols, the adversary performs simple message manipulations: injection, deletion, modification of messages and re-ordering of message sequences. Detailed scenarios of performing such message manipulations are described in [2].

Static model: The honest nodes in the network are denoted by $v_0, \ldots, v_r$, where v_0 denotes the base station, and adversarial nodes are denoted by $v_{r+1}, \ldots, v_{r+m}$. The set of all nodes in the network is denoted by V, and the set of adversarial nodes is denoted by V^*, where $|V| = n = m + r + 1$, and $|V^*| = m$.

The connectivity between nodes is modelled by matrix $\mathbf{E}$, called *reachability matrix*, with size $n \times n$. Here, $\underline{E}_{i,j} = 1$ $(0 \leq i, j \leq n-1)$ if node v_i can communicate with v_j, otherwise $\underline{E}_{i,j} = 0$. We assume that all links are symmetric (i.e., $E_{i,j} = E_{j,i}$ for all i, j).

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). Moreover, the adversary is able to compromise some of these unique identifiers, and all adversarial nodes can use all of those compromised identifiers according to our adversary model.

A *cost function* $\mathcal{C} : V \rightarrow \mathbb{R}$ assigns a cost value, which usually influence the routing decisions, to each node in the network. (e.g., the remaining energy in the battery, the minimal delay of routing messages, or constant 1 to each node in order to represent hop-count, etc.). In our case, the cost function 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 $v^* \in V^*$.

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

Security objective function: Before introducing the security objective function [27] of secure label-switching routing, we introduce some definitions in order to ease its formalization.

Definition 1 (Anchor entry) *An anchor entry* $(v_{src}, v_{dst}, \mathsf{addr}_{nxt}, label^{\mathsf{Out}}_{v_{src}\rightarrow v_{dst}}, \mathsf{delay}_{v_{src},v_{dst}})$ *is the representation of a routing entry at source* v_{src}*, where the destination node is identified by* v_{dst}*), the next-hop towards the destination has (local) address* addr_{nxt}*, the outgoing label of the source towards the destination is* $label^{\mathsf{Out}}_{v_{src}\rightarrow v_{dst}}$*, and the delay of the quickest path through* addr_{nxt} *to the destination is* $\mathsf{delay}_{v_{src},v_{dst}}$.

Definition 2 (Intermediate entry) *An intermediate entry* $(v_{im}, \mathsf{addr}_{nxt}, label^{\mathsf{In}}_{v_{im}\rightarrow v_{dst}}, label^{\mathsf{Out}}_{v_{im}\rightarrow v_{dst}})$ *is the representation of a routing entry at an intermediate node* v_{im}*, where the next-hop towards the destination has (local) address* addr_{nxt}*, the incoming label and the outgoing label of* v_{im} *towards the destination are* $label^{\mathsf{In}}_{v_{im}\rightarrow v_{dst}}$ *and* $label^{\mathsf{Out}}_{v_{im}\rightarrow v_{dst}}$*, respectively.*

Definition 3 (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 4 (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 5 (Workable path) *A sequence of honest nodes* $(v_{\ell_0}, v_{\ell_1}, \ldots, v_{\ell_{d-1}}, v_{\ell_d})$ *is a workable path with respect to configuration* $conf$ *if for all* $0 \leq i \leq d-1$ v_{ℓ_i} *and* $v_{\ell_{i+1}}$ *are direct or pseudo neighbors.*

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

Definition 6 (Correct state) *A state is correct with respect to configuration* $conf$*, if for every anchor entry* $r_0 = (v_{src}, v_{dst}, \mathsf{addr}_{nxt}, label^{\mathsf{Out}}_{v_{src}\rightarrow v_{dst}}, \mathsf{delay}_{v_{src},v_{dst}})$*, where* $v_{src}, v_{dst} \in V \setminus V^*$*, there exists a sequence of intermediate entries* $r_i = (v_{\ell_i}, \mathsf{addr}_{nxt}, label^{\mathsf{In}}_{v_{\ell_i}\rightarrow v_{dst}}, label^{\mathsf{Out}}_{v_{\ell_i}\rightarrow v_{dst}})$ $(1 \leq i \leq d)$ *of honest nodes such that*

- $v_{\ell_d} = v_{dst}$ *and* $label^{\mathsf{Out}}_{v_{dst}\rightarrow v_{dst}}$ *is an application identifier of* v_{dst}*,*
- $(v_{src}, v_{\ell_1}, \ldots, v_{\ell_{d-1}}, v_{dst})$ *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*,*
- $\sum_{j=1}^{d-1} \mathcal{C}(v_{\ell_j}) \leq \mathsf{delay}_{v_{src},v_{dst}}$ *(i.e., the delay of the discovered route between* v_{src} *and* v_{dst} *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 represents the real operation of the network, where each protocol participant is modelled by a probabilistic Turing machine. These machines communicate via common tapes. The specification of each machine and their communication rules are detailed in [27]. Additionally, 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.

We denote the output of the dynamic model by $Out^{\mathcal{F}}_{conf,\mathcal{A}}(z)$, where z is the random input of the model (due to the probabilistic nature of Turing machines). In addition, $Out^{\mathcal{F}}_{conf,\mathcal{A}}$ will denote the random variable describing $Out^{\mathcal{F}}_{conf,\mathcal{A}}(z)$ when z is chosen uniformly at random.

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 7 *A label-switching routing protocol is secure, if for any configuration $conf$ and any adversary $\mathcal{A}$, the probability that $Out^{\mathcal{F}}_{conf,\mathcal{A}}$ equals to zero is a negligible function of κ.*[2]

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.

4.2.2 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 6 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 [4].

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

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 corresponding control message 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.

The third point in Definition 6 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 6, 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.

4.2.3 Proof of Security

Theorem 1 *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 $r_0 = (v_{src}, v_{dst}, \mathsf{addr}_{nxt}, label^{\mathsf{Out}}_{v_{src},v_{dst}}, \mathsf{delay}_{v_{src},v_{dst}})$, but there does not exist a workable path between v_{src} and v_{dst} with $label^{\mathsf{Out}}_{v_{\ell_d} \rightarrow v_{dst}}$ as an application identifier.

- Case 2: There exists an anchor entry $r_0 = (v_{src}, v_{dst}, \mathsf{addr}_{nxt}, label^{\mathsf{Out}}_{v_{src} \rightarrow v_{dst}}, \mathsf{delay}_{v_{src}, v_{dst}})$ and there exists a workable path $(v_{src}, v_{\ell_1}, \ldots, v_{\ell_{d-1}}, v_{dst})$ between v_{src} and v_{dst}, but there does not exist a sequence of intermediate entries $r_i = (v_{\ell_i}, \mathsf{addr}_{nxt}, label^{\mathsf{In}}_{v_{\ell_i} \rightarrow v_{dst}}, label^{\mathsf{Out}}_{v_{\ell_i} \rightarrow v_{dst}})$ $(1 \leq i \leq d)$ 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 $r_0 = (v_{src}, v_{dst}, \mathsf{addr}_{nxt}, label^{\mathsf{Out}}_{v_{src} \rightarrow v_{dst}}, \mathsf{delay}_{v_{src}, v_{dst}})$ and there exists a sequence of intermediate entries $r_i = (v_{\ell_i}, \mathsf{addr}_{nxt}, label^{\mathsf{In}}_{v_{\ell_i} \rightarrow v_{dst}}, label^{\mathsf{Out}}_{v_{\ell_i} \rightarrow v_{dst}})$ $(1 \leq i \leq d)$ where $(v_{src}, v_{\ell_1}, \ldots, v_{\ell_{d-1}}, v_{dst})$ 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 $\sum_{j=1}^{d-1} \mathcal{C}(v_{\ell_j}) > \mathsf{delay}_{v_{src}, v_{dst}}$.

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 (Msg-4) or a RREQ (Msg-3) message with a correct $\mathsf{MAC}_{v_{dst}, v_{src}}$. Let us assume that the adversary $\mathcal{A}$ cannot forge $\mathsf{MAC}_{v_{dst}, v_{src}}$. Thus, $\mathsf{MAC}_{v_{dst}, v_{src}}$ can only be generated by v_{dst} implying that v_{dst} generated and sent a RREQ or RREP message with v_{src} as the destination, and $label^{\mathsf{Out}}_{v_{dst} \rightarrow v_{dst}}$ is an application identifier. Moreover, as $\mathsf{MAC}_{v_{dst}, v_{src}}$ is received by v_{src}, there exists a sequence of nodes $(v_{s_0}, v_{s_1}, \ldots, v_{s_{k-1}}, v_{dst})$ such that $v_{s_{i-1}}, v_{s_i}$ are direct or pseudo neighbors for all $1 \leq i \leq k$, where $v_{src} = v_{s_0}$ and $v_{dst} = v_{s_k}$. This means that there is a workable path between v_{src} and v_{dst} 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 $(v_{\ell_0}, \ldots, v_{\ell_d})$ between v_{src} and v_{dst}, 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 (Msg-4) or a RREQ (Msg-3) message with a correct $\mathsf{MAC}_{v_{dst}, v_{src}}$. Thus, based on Case 1, there exists a workable path $v_{\ell_0}, \ldots, v_{\ell_d}$ between v_{src} and v_{dst} along which the request (or reply) message (Msg-4 or Msg-3), 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 $\mathsf{MAC}^{prv}_{v_{\ell_i}, v_{\ell_{i-1}}}$ can only be generated by v_{ℓ_i}, $v_{\ell_{i-1}}$ received an msg' message ($msg' \neq msg$) with previous-hop MAC $\mathsf{MAC}^{prv}_{v_x, v_{\ell_{i-1}}}$, where $\mathsf{MAC}^{prv}_{v_x, v_{\ell_{i-1}}} \neq \mathsf{MAC}^{prv}_{v_{\ell_i}, v_{\ell_{i-1}}}$. Since $\mathsf{MAC}_{v_{dst}, v_{src}}$ travelled through workable path $(v_{\ell_0}, \ldots, v_{\ell_d})$, v_x is an adversarial node and the adversary obtained $\mathsf{MAC}_{v_{dst}, v_{src}}$ 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, $\mathsf{delay}_{v_{src}, v_{dst}}$ denotes the delay of the travel of $\mathsf{MAC}_{v_{dst}, v_{src}}$ from its originator to v_{src} (either as a part of a RREQ (Msg-3) or a RREP (Msg-4) control message). Let us assume that $\mathsf{MAC}_{v_{dst}, v_{src}}$ cannot be forged by the adversary $\mathcal{A}$. Thus, based on Case 1 and Case 2, $\mathsf{MAC}_{v_{dst}, v_{src}}$ is received on workable path $(v_{\ell_0}, \ldots, v_{\ell_d})$. As the node costs represent the minimum message delays at the nodes and the adversary cannot reduce the delay at the non-corrupted nodes, $\sum_{j=1}^{d-1} \mathcal{C}(v_{\ell_j}) \leq \mathsf{delay}_{v_{src}, v_{dst}}$, 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. ■

5 Conclusion and future work

In this work, 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 (Msg-3), and unicasts the request along with the respective MAC to each neighbor. Although this previous-hop MAC is updated at each hop, the communication and computation costs depend on the number of the neighbors. A reply message (Msg-4) 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.

Finally, we adapted the simulation-based model described in [29] 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. Thus, our future plan is to avoid or mitigate these types of attacks by countermeasures that consider the variety of sensor applications and provide tunable security. In addition, we also plan to implement Secure-TinyLUNAR under TinyOS 2.x [1] and evaluate its performance using TOSSIM [1].

6 Acknowledgements

The work described in this paper is based on results of IST FP6 STREP UbiSec&Sens (`http://www.ist-ubisecsens.org`). UbiSec&Sens receives research funding from the European Community's Sixth Framework Programme. Apart from this, the European Commission has no responsibility for the content of this paper. The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Gergely Ács has been further supported by Ericsson through the HSN Lab.

Levente Buttyán has been further supported by the Hungarian Academy of Sciences through the Bolyai János Research Fellowship.

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] M. Burmester and B. de Medeiros. Towards provable security for route discovery protocols in mobile ad hoc networks, 2007.

[5] CrossBow Technology Inc. MICAz Mote Platform Datasheet. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAZ_Dat 2005.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

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

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.