Securing Topology Maintenance Protocols for Sensor Networks: Attacks and Countermeasures

Andrea Gabrielli and Luigi V. Mancini
Dipartimento di Informatica
Università di Roma "La Sapienza"
00198 Rome, Italy
andrea.gabrielli@gmail.com, mancini@di.uniroma1.it

Sanjeev Setia and Sushil Jajodia
Center for Secure Information Systems
George Mason University
Fairfax, VA 22030
{setia, jajodia}@gmu.edu

Abstract

We analyze the security vulnerabilities of PEAS, ASCENT, and CCP, three well-known topology maintenance protocols for sensor networks. These protocols aim to increase the lifetime of the sensor network by maintaining only a subset of nodes in an active or awake state. The design of these protocols assumes that the sensor nodes will be deployed in a trusted non-adversarial environment, and does not take into account the impact of attacks launched by malicious insider and outsider nodes. We describe three attacks against these protocols that can be used to reduce the lifetime of the sensor network, or to degrade the functionality of the sensor application by reducing the network connectivity and sensing coverage that can be achieved. Further, we describe counter-measures that can be used to increase the robustness of the protocols and make them resilient to such attacks.

1 Introduction

Topology maintenance protocols (TMPs) such as SPAN [3], ASCENT [2], PEAS [5], and CCP [11] are critical to the operation of wireless sensor networks. These protocols aim to increase the lifetime of the sensor network by maintaining only a subset of nodes in an active or awake state, while turning off redundant nodes. The active nodes must be sufficient to maintain the connectivity of the network and also to obtain sensing coverage in the area where the sensor network is deployed.

The various topology maintenance protocols proposed in the literature differ in their objectives as well as the approaches used to achieve their objectives. For example, SPAN and ASCENT aim to maintain network connectivity, but do not guarantee sensing coverage. On the other hand, PEAS and CCP are designed to address both connectivity and the application's coverage requirements in a configurable fashion.

All these protocols involve some form of coordination and message exchanges between neighboring nodes in order to elect coordinators and determine sleep schedules. These protocols were designed assuming a non-adversarial trusted environment. Consequently, they are vulnerable to security attacks in which malicious nodes send spoofed or false messages to their neighbors with the goal of defeating the objectives of the protocol.

Attacks on the topology maintenance protocols can be performed either by entities external to the network (outsider attacks) or by compromised nodes (insider attacks). Insider attacks are a particularly challenging problem for sensor networks because many sensor applications involve deploying nodes in an unattended environment, thus leaving them vulnerable to capture and compromise by an adversary. Unlike outsider attacks, insider attacks cannot be prevented by authentication mechanisms since the adversary knows all the keying material possessed by the compromised nodes.

In this paper, we analyze the security vulnerabilities of three well-known topology maintenance protocols (PEAS, CCP, and ASCENT). We describe three types of attacks that can be launched against these protocols: *sleep deprivation* attacks that increase the energy expenditure of sensor nodes and thus reduce the lifetime of the sensor network, *snooze* attacks that result in inadequate sensing coverage or network connectivity, and *network substitution* attacks in which multiple attackers collude to take control of part of the sensor network.

Further, we describe counter-measures that can be used to increase the robustness of the protocols and make them resilient to such attacks. The proposed counter-measures include authentication mechanisms that can be used to prevent outsider attacks and certain insider attacks (such as impersonation attacks). However, for all these protocols, we

Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'0
0-7695-2369-2/05 $20.00 © 2005 IEEE

found that it is necessary to incorporate protocol-specific measures to increase their resilience to insider attacks. We show that the addition of our countermeasures to the analyzed protocols does not significantly impact their performance or energy consumption.

To the best of our knowledge, the only research work that has pointed out the security issues on topology maintenance protocols is [7]. In particular, Karlof and Wagner [7] describe the *snooze* attack against the following protocols: GAF [13], SPAN [3], CEC [14] and AFECA [12]. However, they do not discuss the use of the snooze attack for reducing the sensing coverage, they do not consider the *sleep deprivation* and *network substitution* attacks, and they do not discuss any counter-measures. In another related work [10], Stajano and Anderson introduced the problem of the *sleep deprivation* attack. However, they did not consider this attack in the context of topology maintenance protocols or describe any countermeasures.

Based on our analysis of PEAS, CCP, and ASCENT, we can make the following general observations with respect to the security considerations in the design of topology maintenance protocols:

- Local broadcast, i.e., a broadcast in which the recipients are restricted to be one-hop neighbors of the sender, is a frequent communication operation used by TMPs. If these locally broadcast messages are not authenticated, TMPs are highly vulnerable to impersonation attacks launched by compromised insiders. Most mechanisms proposed for (global) broadcast authentication appear to be too expensive for resource-constrained sensor nodes. Thus, further research is necessary for developing efficient mechanisms tailored for local broadcast authentication in sensor networks.
- TMPs should be designed so that a node makes its state-transition decisions, e.g., a decision whether to sleep or remain active, based on inputs from multiple neighbor nodes in order to be resilient to false messages injected by malicious nodes.
- TMPs should be designed so that state-transition decisions are revisited periodically. For example without a periodic check of a node's eligibility to be in sleeping or active state, it becomes possible for an adversary to launch a resource-consumption attack that results in a node staying in the active state until its energy is depleted.

The rest of this paper is organized as follows. In Section 2, we discuss the threat model and the different kinds of adversaries we expect to encounter in sensor networks. Next, in Section 3 we present a taxonomy of the attacks that can be launched against topology maintenance protocols. In Sections 4, 5, and 6 we present a brief overview of the PEAS, CCP, and ASCENT protocols respectively and discuss the specific attacks against each protocol. In Section 7 we discuss and evaluate the performance of countermeasures for the three protocols. Finally, in the Appendix we list the notation used in the rest of the paper.

2 Threat Model

In this section, we describe our assumptions with respect to the sensor network and the behavior and capabilities of an adversary.

Because of the wireless nature of communications in sensor networks, we assume that the adversary can eavesdrop on the communications of other nodes and can also inject data packets into the network.

We assume that the nodes are not tamper-proof. Thus if the adversary captures a node, all the information including cryptographic keys stored in the node is compromised. Further, the adversary can clone the identity of a compromised device, storing the information obtained from that node in other malicious nodes.

Finally, we assume that the adversary can deploy malicious nodes, and that these nodes can collude together to attack the system.

2.1 Attacker Classification

We classify the attacker into different categories based on its hardware capabilities, and on the basis of its knowledge of the cryptographic keys used to provide authenticated and/or confidential communication.

Laptop-class vs node-class attackers: A laptop-class attacker uses a relatively powerful device in comparison to a sensor node. An attacker with these capabilities has access to greater battery, storage and computational resources than a typical sensor node, e.g. a Berkeley MICA mote [1]. It may also use high-power radio transmitter and very sensitive antenna that might allow the attacker to eavesdrop on the entire network, and transmit messages with enough power to be heard by any node.

In contrast, a node-class attacker uses one or more devices with the same capabilities as legitimate sensor nodes. Therefore, it is able to listen to or transmit messages only within a limited range, and it faces constraints such as limited battery power, small memory and a relatively slow CPU.

Outsider vs Insider Attackers: An outsider attacker has no more knowledge than the definition of the protocols used in the network and the information gathered by eavesdropping on network communications. It has no access to cryp-

tographic keys or data used to secure the network. For example, it does not possess any credentials that enable it to authenticate itself to other nodes.

In contrast, an insider is an attacker that has all the information used by a node to be a legitimate member of the network, such as its cryptographic keys. It can be a captured node, but also a device, such a node-class or laptop-class, in which the attacker has stored information retrieved from a compromised node.

3 Attacks on Topology Maintenance Protocols

The use of topology maintenance protocols introduces new vulnerabilities in sensor networks. In particular, an adversary can launch new kinds of attacks exploiting the ability of these protocols to increase or decrease the number of active nodes. In the following discussion, we present three different attacks on topology maintenance protocols.

Sleep Deprivation Attack: In this attack, the adversary tries to induce a node in a specific area to remain active. This attack has two effects. First, by increasing the energy expenditure of sensor nodes, it reduces the estimated lifetime of the network. Second, in the case of a densely populated area, it can result in increased energy consumption due to congestion and contention at the data link layer.

Snooze Attack: In this attack, the adversary forces nodes to remain in the sleeping state. This attack can be applied to the whole network or to a subset of nodes. In the second case, the adversary can launch an attack to jeopardize the connectivity of the network or to reduce the sensing coverage in a region. For example, an adversary can selectively turn off nodes that are monitoring an intruder's path through an area in which a sensor field has been deployed for surveillance.

Network Substitution Attack: In this attack, the adversary takes control of the entire network or a portion of it using a set of *colluding* malicious nodes. The adversary deploys a set of nodes that are included in the set elected by the topology maintenance protocol to maintain the network connectivity or the sensing of the area. Once the protocol has chosen the malicious nodes as working nodes, the portion of the network under attack is totally in the hands of the adversary.

When the adversary controls a portion of the network, it can perform other attacks such as traffic analysis and selective or complete packet dropping. This attack cannot be easily detected because the adversary can maintain the network connectivity and keep it operating normally. For example, if the application is supposed to receive readings from sensors at a certain frequency, the adversary can send false readings at the same rate and avoid detection.

4 Analysis of PEAS

4.1 Brief Review of PEAS

In PEAS [5], the subset of active nodes selected by the protocol must be sufficient to obtain sensing coverage of the deployment area and sufficient to maintain the connectivity of the network. The operation of the protocol is described below.

Each node in PEAS has three operation modes: *Sleeping*, *Probing*, and *Working*. In the *Sleeping* mode, a sensor is not active. It sleeps for an exponentially distributed duration $f(T_s) = \lambda e^{-\lambda T_s}$, where λ is the probing rate of the node and T_s denotes the sleeping time duration. A node in *Sleeping* mode waits until its sleeping time expires and then enters the *Probing* mode. In the *Probing* mode a sensor tries to detect whether any working node is present within a probing range R_P. The probing node sends a PROBE message with the range R_P and any working node within R_P should respond with a REPLY message, also sent within the range of R_P. If the probing node receives a REPLY, it goes back to *Sleeping* mode, otherwise, it enters the *Working* mode. Normally, a node stays in the *Working* mode until it consumes all its energy. However, as discussed below, in some situations, a node is allowed the transition back to the *Sleeping* mode.

PEAS provides a mechanism to adjust the probing frequency of each sleeping node and maintain the probing rate of all the neighbors at a desired value λ_d. A working node measures the probing rate $\hat{\lambda}$ of its neighbors and includes that measure in its REPLY messages in response to probes. When a probing node receives more than one REPLY, it selects the largest $\hat{\lambda}$ and sets its new probing rate λ^{new} as follows:

$$\lambda^{new} = \lambda^{old} \frac{\lambda_d}{\hat{\lambda}}$$

In this way, the protocol keeps the actual probing around the desired λ_d. Note that the sleeping time of a node increases with $\hat{\lambda}$.

Due to collisions, PROBE and REPLY messages can be lost and some nodes could become active even if they are inside the probing range of another working node. For example, assume that two active nodes *u* and *v* are within each other's probing range. If a third node *z* that is within the probing range of *u* and *v* sends a PROBE, then both *u* and *v* will overhear each other's REPLY. In this case, PEAS allows a node to go back to sleep, using the following rule: if a node *u* hears a REPLY by a node *v* that is active and with a working time T_w that is greater than the working time of

u, then *u* goes back to sleep. To allow a node to determine if its working time is greater than the working time of another node, each node must include the working time in its REPLY message.

4.2 Attacks on PEAS

PEAS is vulnerable to all the attacks presented in Section 3. In the following discussion, we present five attacks on PEAS. The first three attacks are based on the use of forged REPLY messages, whereas the last two *sleep deprivation* attacks exploit the fact that in PEAS, once a node enters the working state, it will normally remain active until it runs out of power.

Snooze attack by a Laptop-class attacker The adversary *X* sends a forged REPLY with enough transmit power to be heard by any node of the network. In the forged REPLY, T_w is set to the maximum value it can assume. Each working node goes to sleep because it believes that there is another working node within its probing range and with a greater T_w. Further, the adversary can use the $\hat{\lambda}$ value included in the REPLY message to control the sleep schedules of other nodes. For example, he can set a small $\hat{\lambda}$ to make other nodes wake up very often and rapidly consume their energy. To disable the network during selected periods of time, he can use a large value for $\hat{\lambda}$ and thus induce other nodes to sleep for a long time.

Snooze attack by node-class attacker The adversary uses a set of nodes to turn off the network in a selected area. It partitions the target area into cells C_1, C_2,..., C_n, where each cell is a square of size $\frac{2}{\sqrt{2}}R_T$. This choice ensures that a node in the center of a cell C_i can cover the cell with its transmission range. Then, the adversary deploys its malicious nodes in the area with a density such that it has a device inside each cell. Each node has the task to keep legitimate nodes that are within its transmission range in the sleeping state. When the attack begins, a malicious node monitors the network for incoming messages. If it receives a probe message or discovers an active node, it sends a REPLY message with T_w set to the maximum value it can assume to put the node to sleep. Further, the $\hat{\lambda}$ in each REPLY is chosen in a way that the neighbors are synchronized to wake up and probe together. This allows the adversary to wake up at the same time as its neighbors, wait until all of them have sent probe messages, and then send a single REPLY message that results in all active nodes going to sleep. In this way, it is possible for the adversary to consume the same amount of battery power as any of its neighbors and to ensure that it will not run out of power before its neighbors.

Network substitution attack The adversary substitutes legitimate nodes in a portion of the network with malicious nodes. The attack is launched in the same fashion as the *snooze attack by node-class attacker*, but now the size of the cells is $\frac{R_T}{\sqrt{5}}$. In this way, any two nodes in adjacent cells or in the same cell are within each other's transmission range. With a node inside each cell, the adversary can keep any legitimate node in sleeping mode and at the same time can maintain the connectivity between the malicious nodes and between the malicious nodes and the rest of the network.

Sleep Deprivation attack by a Laptop-class attacker The goal of this attack is to keep all nodes in *Working* mode. Following the same procedure used in the *snooze attack by a laptop-class attacker*, the adversary can put all the nodes in sleeping mode and synchronize them so that they all wake up at a given time. When all the nodes wake up to probe, the attacker jams the network to prevent any of them from receiving a REPLY. All the nodes now are in *Working* mode and no one will probe again. Even if two nodes are within each other's probing range, they will not go back to sleep because they will not receive any REPLY.

Sleep Deprivation attack by node-class attacker The adversary deploys a set of malicious nodes in the area under attack and induces all the legitimate nodes to transition to *Working* mode at the same time. Note that the adversary cannot leave even a single node in *Sleeping* mode, because that node could set off a chain reaction that results in all the redundant active nodes returning to the *Sleeping* mode. This is because when a sleeping node wakes up and sends a PROBE message, any active node within the probing range will respond with a REPLY message. The REPLY will be heard by a redundant active node resulting in its going back to sleep.

The adversary partitions the target area into squared cells C_1, C_2,..., C_n, where each cell is a square of size $\frac{R_T}{\sqrt{5}}$. In this way, any two nodes in adjacent cells or in the same cell are within each other's transmission range. The adversary applies the attack in the same way as the *network substitution attack* and puts all the nodes in *sleeping* mode. In particular, by including in each REPLY an appropriate $\hat{\lambda}$, the sleep phases of all the legitimate nodes are synchronized so that they wake up and probe at the same time. At this time, the malicious nodes jam the network. Hence, any legitimate node goes into *working* mode because it has not received any REPLY messages.

5 Analysis of CCP

5.1 Brief Review of CCP

The goal of CCP [11] is to maintain an active subset of the network nodes that is sufficient to guarantee a K_S-coverage degree of the network deployment area. In CCP each node decides locally to be active or to go to sleep using the K_S-coverage eligibility rule and the information received in the HELLO messages sent by its neighboring nodes.

Before introducing the K_S-coverage eligibility rule, we give some definitions. The *sensing circle* of a node *u* is the set of points *p* that are R_S distant from *u*. A point is called an *intersection point* if it is the intersection between two sensing circles or if it is the intersection between the sensing circle of a node and the boundary of the area to be covered.

The K_S-coverage eligibility rule is that a node *u* is not eligible to become active if: (1) any point *p* that is an *intersection point* between the sensing circles of two active nodes and that is also within the sensing circle of *u* is K_S-covered, or (2) there is no intersection point inside the sensing circle of *u* and there are at least K_S sensors that are located at the same position as *u*. The protocol ensures that if a node *u* decides to sleep, then each point *P* that could be sensed by *u* is already sensed by at least K_S other nodes. In the following discussion, we summarize how the protocol works.

The broadcasting of HELLO messages allows all the nodes to know the positions of the active nodes and decide whether to remain active or to go back to sleep. When $R_T \geq 2R_S$ a node includes in its HELLO message only its own location. When $R_T < 2R_S$, a node may have inside its *sensing circle*, intersection points of nodes that are $\lceil \frac{2R_S}{R_T} \rceil$ hops away. Hence, in this case the node should include the locations of all the active node within $\lceil \frac{2R_S}{R_T} \rceil$ hops, otherwise more nodes will remain active. A node discovers the position of active nodes within $\lceil \frac{2R_S}{R_T} \rceil$ thanks to the HELLO messages received from the neighbors.

A node can be in one of three states:

- SLEEP: in this state a node sleeps until its sleep timer T_s expires. Then it wakes up and enters the LISTEN state.
- LISTEN: in this state, the node collects beacon messages, i.e., locally broadcast HELLO, WITHDRAW and JOIN messages, and executes the K_S-coverage eligibility rule. If the node is eligible, it enters the ACTIVE state and broadcasts a JOIN message, otherwise it goes back to the SLEEP state.
- ACTIVE: In this state, each time a node receives a beacon message, it executes the K_S-coverage eligibility rule to determine its eligibility. If it is ineligible it sends a WITHDRAW message and goes back to sleep.

Both the join and withdraw announcements are delayed by randomized timers to avoid collisions. After the random delay expires, a node checks its eligibility status and if the eligibility status has changed, the node cancels the announcement and remains in its previous state. This prevents a situation in which two nodes both announce that they cover the same portion of network when just one of them is sufficient and also avoids situations in which two nodes withdraw at the same time leaving a zone with a coverage less then K_S.

5.2 Attacks on CCP

We now describe three attacks on CCP in which the adversary exploits the possibility of using forged HELLO messages to keep nodes in the sleeping mode. We also discuss why the design of CCP makes it difficult to launch a sleep deprivation attack.

Snooze attack by a Laptop-class attacker The goal of the attacker is to induce all the legitimate nodes inside a selected area *B* of the network to go into the sleep mode. For the sake of simplicity, we first describe the attack assuming that the required coverage degree K_S of the network is 1 and $R_T \geq 2R_S$.

The adversary selects an area $B' \supseteq B$ with the following properties: (1) it is a composition of cells C_1, C_2,..., C_m, where the size of a cell is less than $\frac{2}{\sqrt{2}}R_S$, and (2) it contains all the sensing circles of the target nodes. The size ($\frac{2}{\sqrt{2}}R_S$) of the cell ensures that any point inside or on the boundary of a cell is within a circle of radius R_S centered in the cell, and the second property ensures that all the *intersection points* of the target nodes are within *B'*. For each cell C_i, the adversary chooses a bogus node ID_i and broadcasts a HELLO message with enough transmit power to be heard by every node in *B'*, announcing that ID_i is active and positioned at the center of the cell C_i. In this way any node inside C_i believes that its *sensing range* inside *B'* is covered by the bogus nodes. All the target nodes remain sleeping because they are within *B'* and all their *intersection points* are also within *B'*.

To launch the attack when the required coverage degree of the network is a general $K_S > 1$, the adversary has to choose K_S bogus node identifiers for each cell C_i.

When $R_T < 2R_S$, for each bogus node ID_i the adversary has to include in the HELLO message of node ID_i all the positions of the bogus nodes that are $\lceil \frac{2R_S}{R_T} \rceil$ hops away from ID_i.

Snooze attack by node-class attacker In this attack, the adversary uses malicious nodes to keep all the nodes inside a selected area *B* of the network in the sleeping state. We describe the attack assuming that $K_S = 1$, and then we explain how to extend it for a general K_S.

The adversary calculates the minimum number of nodes *m* necessary for sensing a circular area C of radius R_T+R_S, and their relative positions RP_1, RP_2,..., RP_m with respect to the center of the area C. He loads positions RP_1, RP_2,..., RP_m and a set of *m IDs* into a single malicious node. Then the attacker deploys the malicious nodes in *B* in a way that any point inside *B* is within the communication range of a malicious node.

The task of a malicious node *X* is to make other legitimate nodes inside its transmission range think that they are ineligible. Note that malicious node X can announce, to nodes within R_T, the existence of fictitious nodes in the circular area of radius $R_T + R_S$. In this way even the legitimate nodes near the border of the area of radius R_T will believe that their sensing range is already covered. Each node *X*, using its absolute position, calculates the absolute position AP_1, AP_2,..., AP_m of RP_1, RP_2,..., RP_m. Then it sends *m* HELLO messages announcing that ID_i ($i = 1$ to *m*) is active and located at AP_i. Each neighbor node of *X* believes that all the points inside its sensing circle are already covered by one of the nodes in the set $\{ID_1, ID_2, \ldots, ID_m\}$ because its sensing circle is within $R_T + R_S$ from *X*. When a legitimate node that is a neighbor of *X* runs the K_S-coverage eligibility rule, it finds that all the intersection points within its sensing circle are covered, so it believes itself to be ineligible. In this way, the attacker induces all the legitimate nodes inside *B* to go to sleep because any legitimate node is a neighbor of at least one malicious node. When $R_T < 2R_S$ a malicious node can include in its HELLO message the position of bogus active nodes within $\lceil \frac{2R_S}{R_T} \rceil$ hops.

The attack can be extended to a generic K_S by announcing K_S nodes at each position AP_1, AP_2,..., AP_m.

Network substitution The attack is performed in the same manner as the previous *snooze attack by node-class attacker* but the adversary partitions the target area into squared cells C_1, C_2,..., C_n, where each cell is a square of size $\frac{R_T}{\sqrt{5}}$. In this way, any two nodes in adjacent cells or in the same cell are within each other's transmission range. Thus the malicious nodes are connected and they can simulate the normal operation of the network.

Sleep Deprivation attack To launch this attack, the adversary has to jam the network to prevent the reception of the beacon messages periodically exchanged by the nodes. Further, due to the continuous checking of the eligibility status by the active nodes, this operation must be performed very often, making the attack not affordable for node-class attackers, and also expensive for a laptop-class attacker. Moreover, note that a denial-of-service attack involving continuous jamming can be performed in any sensor network, irrespective of the topology maintenance protocol being used. Hence, we do not consider this attack as an attack specific to topology maintenance protocols.

6 Analysis of ASCENT

6.1 Brief Review of ASCENT

ASCENT [2] adaptively elects a set of active nodes that stay awake all the time and perform multi-hop packet routing, whereas the rest of the nodes remain passive and periodically check if they should become active. Each node decides to be active or passive using a local measurement of its connectivity degree (number of active neighbors) and a measurement of its data loss (DL) rate.

The protocol is based on the idea that the DL rate of the network should not be higher than the specified loss threshold (LT), and the number of active nodes in a communication range should not exceed the neighbor threshold (NT). A node is considered as an active neighbor if its neighbor link loss is under a specified threshold. Each node adds a unitary monotonically increasing sequence number to each data and control packet transmitted. This permits neighbor link loss detection when a sequence number is skipped. Similarly, a separate sequence number is used to detect lost application data packets. Note that the data loss rate (DL) is estimated based on application data packets, and control packets are not considered in its calculation.

Nodes can be in one of four states: ACTIVE, SLEEP, PASSIVE, or TEST:

- ACTIVE: The node works until its energy is depleted. If it measures a DL rate greater than LT, the active node broadcasts a HELP MESSAGE to its neighbors.
- SLEEP: In this state, the node turns off its radio and sleeps for a time T_s. When T_s expires, the node moves into the PASSIVE state.
- PASSIVE: The intuition behind the PASSIVE state is to collect information regarding the state of the network without causing interference with other nodes. The passive node turns its radio on and sets the network interface in promiscuous mode to overhear all the packets transmitted by the neighbors. When a node enters this state, it sets up a timer T_p and sends a NEW PASSIVE NODE ANNOUNCEMENT message. This message is used by active nodes to estimate the density of the nodes in the neighborhood. Active nodes transmit this density estimate to any new passive node.

When T_p expires, the passive node enters the SLEEP state. Before T_p expires, if the number of active neighbors is below NT and either the data loss DL is higher than the loss threshold LT or DL is below the LT but the node received a HELP MESSAGE from an active neighbor, then the passive node enters the TEST state.

- TEST STATE: A node in this state probes the network to see if the addition of itself may improve connectivity. The test node starts exchanging data and routing control messages with its neighbors. It sets up a timer T_t and sends a NEIGHBOR ANNOUNCEMENT message. If a node in TEST state receives a NEIGHBOR ANNOUNCEMENT message from a node with an higher ID, then it goes back to PASSIVE state. When T_t expires, the node enters the ACTIVE state. If before T_t expires, the number of active neighbors is above NT or the average DL rate is higher than the average DL rate before the node entered the TEST state, then the node moves into the PASSIVE state.

The timers T_p and T_t are fixed. However, the value of T_s is dynamically adjusted using the estimated node density in the neighborhood. Specifically, it increases with the node density.

6.2 Attacks on ASCENT

We now describe *Snooze*, *Network Substitution*, and *Sleep Deprivation* attacks on ASCENT. In ASCENT a node that enters the active state does not go back to sleep for any reason, so an attacker is not able to put it in sleeping mode. The snooze attacks that we describe aim to maintain in PASSIVE or in SLEEP state nodes that are not in ACTIVE state. These attacks begin to be effective only when the initial set of active nodes fail. This is because those nodes will not be replaced by new nodes leaving the area of the network that is under attack without a sufficient number of active nodes. For this reason, if the attacker wants to disable the network at time *T*, he has to start the attack before *T* and wait for active nodes in the area to start running out of battery power. Thus, in contrast to the *snooze* attacks presented for CCP and PEAS, the attacker cannot simply launch the attack at the moment he wants to turn off the network, because some legitimate nodes will continue to work. Further, it can be difficult to choose when to start the attack to disable the network at time *T*, because the attacker may not be able to estimate the remaining battery power of the nodes that are in active state.

Snooze attack using impersonation The adversary impersonates multiple active nodes so that the legitimate nodes estimate that the number of active neighbors is greater than NT. Thus, all the nodes that are in TEST state enter the PASSIVE state, and all the nodes in PASSIVE state transition to the SLEEP state. Although the adversary cannot affect nodes in ACTIVE state, when these nodes fail they are not replaced by new nodes, and this will cause an incremental degradation of the connectivity and sensing coverage of the network.

This attack is more effective when launched by a *laptop-class attacker*. In fact, a laptop-class adversary has to impersonate only NT identities to launch the attack against all the nodes of the network. However, a node-class adversary has to impersonate NT nodes to launch an attack against just its neighbors.

Snooze attack using NEIGHBOR ANNOUNCEMENT messages The adversary periodically broadcasts a NEIGHBOR ANNOUNCEMENT message with an *ID* that is higher than the identifiers of all the legitimate nodes of the network. This induces all the nodes that are in TEST state to think that they have a neighbor with an higher *ID* in TEST mode. Each legitimate node in TEST state will go to the SLEEP mode. In this way the adversary can keep all the nodes that are not already active in the SLEEP state. Thus, when active nodes fail they are not replaced by new nodes causing an incremental degradation of the connectivity and sensing of the network.

This attack can be launched either by a *laptop-class attacker* or a *node-class attacker*. In the first case, the adversary can attack the entire network by simply sending the message periodically all over the network. In the second case, the malicious node can launch the attack against all its neighbors. To minimize its energy consumption, the malicious node can wait for the NEIGHBOR ANNOUNCEMENT message of a neighbor, and then send a NEIGHBOR ANNOUNCEMENT message with an higher *ID* to make the node go back to sleep. The adversary can deploy a set of malicious nodes to launch the attack in a selected part of the network and to block communication or sensing coverage in the target area.

Both of the previous *Snooze* attacks can be made more effective by increasing the time interval for which a legitimate node remains in SLEEP state. In ASCENT, a node adjusts the timer T_s based on the neighbor density estimates received from active nodes, and it increases with the node density. The attacker can send false NEW PASSIVE NODE ANNOUNCEMENT messages with different *ID*s to leading active nodes to increase their estimates of the node density, or it can directly announce a high node density to the other nodes.

Network substitution attack In this attack, the adversary has to deploy a sufficient number of nodes so that they can form a connected network and so that any legitimate node

in the area being attacked is in the transmission range of at least one malicious node. After their deployment, all the malicious nodes launch a *snooze attack using NEIGHBOR ANNOUNCEMENT messages*. When the currently active nodes run out of power, the adversary controls the part of the network attacked, because no new nodes will enter the TEST or ACTIVE states. The only services available in the attacked area are the ones provided by the malicious nodes.

Sleep Deprivation attack In normal operation, the number of active nodes selected by ASCENT is just sufficient to maintain the data loss rate under a threshold LT specified by the application. However, the adversary can use HELP messages to increase the number of active nodes and reduce the efficiency of ASCENT as follows.

First, the adversary simulates an increase in the data loss rate by sending to u several messages with sequence numbers that differ by more than 1. This leads node u to compute an erroneous estimate of the data loss rate. The adversary then sends a HELP message and if a node u has less than NT active neighbors than it will transition to the TEST state. Once u enters the TEST state, the adversary can manipulate the sequence numbers of messages it sends to convince u that the data loss rate has decreased. Consequently, u will then transition to the active state.

If u has more than NT active neighbors, the adversary has to simulate first a decrease in the number of active neighbors of u, and then the adversary can carry on the attack on the data loss rate explained above to bring u to the ACTIVE state. To simulate a decrease in the number of active neighbors of u, the adversary starts sending to u a sequence of messages where the adversary maliciously skips few numbers in the sequence every time a message is sent to simulate an increase in the neighbor link loss. In particular, the adversary forges the sender identifier of each message to convince u that many links of are lossy. At this point u, following the ASCENT protocol, starts decreasing the number of its active neighbors due to the increase in the measured neighbor link loss. In particular, when the number of active neighbors is less than NT, the adversary can convince u to go in ACTIVE state by repeating the attack explained above.

In this way the adversary can induce most of his neighbors to transition to the ACTIVE state. This attack can be performed either by a *laptop-class attacker* or a *node-class attacker*. Even if the adversary has to apply the attack to his neighbors one by one, the adversary does not need to repeat the attack periodically because nodes in ACTIVE state do not go back to sleep. Thus, the adversary incurs an energy expenditure only once for each node that is attacked.

7 Countermeasures

We now describe countermeasures for each protocol we have analyzed. Clearly, the most important countermeasure against the attacks described in Section 3 is to ensure that all communication between nodes is authenticated. Many of the messages exchanged between nodes in the TMPs are local broadcast messages, i.e. broadcasts that are restricted to the immediate neighbors of a node. Thus, efficient authentication mechanisms are needed not just for unicast messages but also for local broadcast messages.

Several solutions have been proposed in the literature for efficient key setups in sensor networks [9], [15], [4]. In our discussion below, we assume that neighboring nodes can establish pairwise shared keys with each other. The pairwise shared key is used for computing message authentication codes (MACs) for authenticating unicast messages exchanged between two neighboring nodes. We note that ideally the pairwise key establishment scheme should be resilient to cloning or node replication attacks. In particular, we require that even if the adversary captures a node *u*, the identity of the compromised node cannot be successfully impersonated outside the neighborhood of *u*. One possible protocol for achieving this goal in a sensor network is LEAP [15]; however, any scheme that meets these requirements can be used.

In addition, we assume the existence of an authentication mechanism for local broadcast authentication. In particular, we use the mechanism based on one-way key chains provided by LEAP for authenticating local broadcasts. However, other protocols could be used, e.g., μTESLA [9], for authenticating the locally broadcast messages. Alternatively, new local broadcast authentication mechanisms could be designed that are more efficient and scalable than the existing solutions.

In the following discussion, we do not consider attacks which consist of jamming the network to produce a denial-of-service attack since these attacks are not specific to topology maintenance protocols.

Before discussing our countermeasures, we briefly describe the LEAP [15] key management protocol for sensor networks. It allows a node *u* to share a pairwise key K_{uv} with any neighbor *v*. Further, LEAP restricts the security impact of a node compromise to the immediate network neighborhood of the compromised node. This provides two useful properties. First, if a node *u* is captured, the adversary cannot use the compromised keys obtained by *u* to create new pairwise keys. Second, the keys of a node *u* are established between *u* and its neighbors after the deployment of *u*, and *u* can use that keys only with them. Thus, an attacker that compromises *u* cannot use its compromised keys in other parts of the network; hence it cannot exploit the possibility to clone the identity of *u*.

Further, LEAP provides a local broadcast authentication mechanism based on the use of a one-way key chain. The LEAP uses one-way key chain [8] for one-hop broadcast authentication. Unlike TESLA, this technique does not use delayed key disclosure and does not require time synchronization between neighboring nodes. Basically, every node generates a one-way key chain of certain length, then transmits the commitment (i.e., the last key generated) of the key chain to each neighbor, encrypted with their pairwise shared key. Whenever a node has a message to send, it attaches to the message the next K_{u_j} key in the key chain. The keys are disclosed in an order reverse to their generation. A receiving neighbor can verify the first message authenticated with the key-chain based on the commitment, and it can verify the following messages based on the key it received from the sending node more recently. As discussed by the authors [15], this authentication mechanism has some weaknesses and it is possible for an adversary to obtain a valid key for authenticating a fabricated message while impersonating another node. Specifically, in this attack, an adversary has to receive the original broadcast message from a node (say S) to obtain the valid authentication key (say K), while at the same time preventing the target node(s) from receiving the broadcast message. It can then impersonate S by including the key K in a fabricated message that it sends to the target node(s). However, we show that this attack does not give any real advantage to the adversary in the case of attacks on TMPs.

7.1 Proposed countermeasure for PEAS

We present an extension of the PEAS protocol that makes the protocol resilient to outsider attackers. Further, a node can tolerate attacks by up to *t* compromised nodes within its transmission range R_T. We assume that the network nodes have established pairwise keys with their neighbors.

We extend the PEAS protocol in the following way. In *Probing* mode any node, say *u*, sends a PROBE containing a *nonce* and its *ID*. If *u* receives at least *t+1* authenticated REPLYs with the corresponding *nonce*, it goes back to sleep. Each REPLY, sent by a neighbor *v* of *u*, must include a MAC created with the shared pairwise key K_{uv}. The messages exchanged between *u* and *v* are:

$$\begin{aligned} u \longrightarrow * \quad &: \quad PROBE, u, nonce \\ v \longrightarrow * \quad &: \quad REPLY, v, \hat{\lambda}, T_w, \\ &\quad\; MAC(K_{uv}, REPLY|nonce|\hat{\lambda}|T_w) \end{aligned}$$

The use of the *nonce* is introduced to avoid a replay attack. Without the *nonce* an adversary can store the REPLY from *v* and uses it to respond to *u* pretending that *v* is functioning even when *v* is dead. With such a replay attack the adversary could prevent new legitimate nodes from becoming active.

In the original protocol a working node goes back to sleep if it overhears a REPLY from a node (in response to a PROBE) with a larger working time T_w. This allows nodes that became active erroneously (e.g., because the REPLY messages from active nodes were lost due to collisions) to go back to the sleep state. We maintain this feature but we modify the rule as follows: a working node goes back to sleep only if it receives $t + 1$ REPLYs. Further, when this situation occurs, if the node has the smallest T_w then it performs a probe by sending a PROBE message. If it receives at least *t+1* REPLYs in response to the probe, it goes back to sleep. This extra probing operation is performed by the node to authenticate and also verify the freshness of the REPLY. Note that even if a node *z* receives the REPLY sent by a node *v* to *u*, node *z* cannot check the freshness, because *z* does not know the *nonce*, and *z* cannot verify the authenticity of the message, because the message is authenticated with a key shared only between *u* and *v*.

With these modifications, an adversary has to compromise at least *t+1* nodes within the transmission range of *u* to be able to induce node u to go to sleep. If LEAP is used to establish the pairwise keys, the adversary cannot reuse the compromised identities at different locations in the network. Thus, the number of nodes that need to be compromised in order for the adversary to be able to launch the attacks described in Section 4.2 increases with the value *t* and the size of the target area, and we can choose *t* to achieve the desired resilience to node compromise.

7.2 Proposed countermeasure for CCP

We present an extension of the CCP protocol that makes it resilient to outsider attackers. Further, a node can tolerate attacks by up to $K_S - 1$ compromised nodes within its transmission range R_T. We assume that nodes use the LEAP protocol to establish pairwise keys with their neighbors, and for local broadcast authentication.

We extend the CCP protocol in the following way. All the beacon messages broadcast by a node *u* are authenticated using the mechanism provided by LEAP. In other words, each message includes both a MAC and the key used for computing the MAC. (Here the key is part of a one-way key chain maintained by each node. The keys in this key chain are revealed in the reverse order of their generation.)

Further, each node includes in the HELLO message only its own location, so that a node can lie only about its position. With this modification, when $R_T < 2R_S$ the protocol may elect a few more active nodes than are necessary. This may not necessarily be negative since it increases the delivery ratio of the network, as shown in [11].

For example, the HELLO message that *u* sends to the neighbors using the j-th key of the chain would have the

format:

$$u \longrightarrow * \quad : \quad HELLO, u, K_{u\_j}, pos, \\ MAC(K_{u\_j}, HELLO|pos)$$

When a node *v* receives a beacon message, it verifies the validity of the new key using the previous key received from *u*. If the key or the MAC are not valid then *v* discards the message, because it could have been sent or modified by a malicious node. The sequentiality of the keys also prevents replay attacks in which an adversary replays the HELLO messages sent by legitimate nodes when these nodes have failed or are in an inactive state.

The vulnerability of LEAP's one-way key chain mechanism discussed in [15] does not provide any real advantage to the adversary in the case of CCP. In order to turn off a legitimate node v, the adversary has to perform the attack on LEAP's local broadcast authentication mechanism for K_S different neighbors of v (i.e., the adversary has to intercept the messages and modify maliciously the position fields of K_S nodes). Moreover, the only field the adversary can change is the position field. In stationary sensor networks, this change can easily be detected.

Note that the minimum number of identities needed to keep a node in the sleep state is K_S. Thus, with our extension to CCP, an adversary has to compromise at least K_S nodes within the transmission range to launch a snooze attack on *u*. If LEAP is used, the adversary cannot successfully use compromised identities at different locations in the network.

The number of nodes that have to be compromised by the adversary in order to launch the attacks described in Section 5.2 increases with the value K_S and the size of the target area. We can choose K_S to achieve the desired resilience to node compromise.

7.3 Proposed countermeasure for ASCENT

We present some countermeasures to make the ASCENT protocol resilient to outsider attackers and to improve the resilience against insiders. We assume that nodes use the LEAP protocol to establish pairwise keys with their neighbors and for local broadcast authentication.

First, metrics used by the protocol such as the node density should be computed based on estimates provided by multiple nodes in order to be robust to false estimates supplied by a malicious node.

Second, the mechanism to estimate the DL rate and the neighbor link loss using sequence numbers should be designed in a way that the adversary is not able to trick the mechanism. One solution could be that each node uses a sequence number window that defines the interval of sequence numbers accepted. This prevents a node from accepting a message with a large sequence number that makes the estimated DL rate very high.

Third, HELP messages are authenticated using LEAP's local broadcast authentication mechanism. Thus, a malicious node cannot send HELP messages while impersonating another node. Further, we can define a number *h* of HELP MESSAGEs that a node *u* must receive from *h* different nodes before *u* tries to join the network. In this way. we can increase the resilience against malicious insider nodes that want to increase the number of active nodes.

To prevent a malicious node from keeping nodes in sleeping mode using a NEIGHBOR ANNOUNCEMENT MESSAGE with an high ID, we can use the following modification. Each node *v* stores a sequence number na_seq_i for each neighbor *i* to count the times it receives a NEIGHBOR ANNOUNCEMENT MESSAGE from the neighbor *i*.

Node v includes in the NEIGHBOR ANNOUNCEMENT MESSAGE the MAC computed with the next key of the one-way key chain, and the key used for the MAC. In addition, a node u maintains a sequence number na_seq_u counting the number of times u sends a NEIGHBOR ANNOUNCEMENT message (i.e., the number of times u enters the TEST state).

When node u in TEST state receives a NEIGHBOR ANNOUNCEMENT message from v, node u checks that the key inside the message is a valid key of v's one-way key chain, and that the MAC is computed with that key. If the key is not valid or the MAC is not correct, u discards the message, otherwise node u increments na_seq_v by 1 and changes state using the following rules:

- if na_seq_v is greater then na_seq_u then *u* ignores the message, remaining in the TEST state.
- if na_seq_v is less then na_seq_u then *u* makes a transition to the PASSIVE state.
- if na_seq_v is equal to na_seq_u then if the ID of *v* is smaller than the ID of *u*, *u* discards the message, otherwise it makes a transition to the PASSIVE state.

In other words, if multiple nodes make a transition to the TEST state, the tie-breaking mechanism is based on the number of transitions to TEST state.

As a final preventive measure, we can use different one-way key chains for the HELP and NEIGHBOR ANNOUNCEMENT messages to make the vulnerability of the one-way key chain useless for the adversary. In fact, in this way a single key-chain is used for encoding the type (and content) of the message. Hence, even if the adversary obtains a valid key in the key chain, the adversary can use that key only to send the same identical message.

7.4 Performance evaluation

The performance overhead of the countermeasures discussed above arises from two sources. First, we require all messages to be authenticated, and this leads to both computational overhead (for computing and verifying MACs) and communication overhead (due to the increased message sizes). Second, some of our countermeasures involve modifications to the TMPs that lead to increased storage costs per node (for maintaining additional state information) and increased communication overhead.

In the following discussion, we analyze the authentication overhead as well as the overhead for the protocol specific countermeasures. Note that we have not introduced new messages in any protocol. Thus our overhead is only due to the authentication of messages, in some additional storage cost, and in an increased communication overhead.

7.4.1 Authentication Overhead

With respect to the overhead due to the addition of authentication, we note that there is no additional authentication requirement imposed by our countermeasures beyond what is already recommended for preventing outsider attacks. Authentication is necessary not just for securing topology maintenance protocols but for securing any communication in sensor networks.

Our countermeasures rely on unicast messages being authenticated using MACs, and local broadcast messages being authenticated using the technique proposed in LEAP. We note that the local broadcast authentication mechanism proposed by LEAP involves attaching a single MAC and its key to each broadcast message. The TinySec [6] project has demonstrated that the addition of a MAC adds less than 3% to the energy expenditure of sending a packet and less than 1.6% to the latency of packet transmission.

We think that these costs are reasonable, and should not be considered as an overhead specifically for securing topology maintenance protocols, but instead as the cost of securing the communication in a sensor network.

7.4.2 Storage Costs

For all the protocols, the main storage cost imposed by our countermeasures is for the storage required for pairwise keys and the one-way key chain used for local broadcasts. In [15], Zhu et al analyze the storage requirements for LEAP and show that it is reasonable even for the current generation of sensor nodes.

PEAS The additional state information that a node needs to maintain for our modifications to PEAS is the following: (i) a counter and a list of IDs for the received REPLY messages, (ii) a nonce to check the freshness of the REPLY. The actual storage overhead would depend on the implementation. For example, most routing or link layer protocols use a neighbor table. Thus in the event we already have such a table we do not need to store the ID of the node that sent the REPLY; we can simply add a flag to each entry in the neighbor table that is set when a REPLY message is received from that node.

CCP There is no additional storage cost for our modifications to CCP beyond the cost of maintaining keys for authentication.

ASCENT The additional state information that a node needs to maintain for our modifications to ASCENT is the following: (i) a counter to keep track of the number of HELP requests received (ii) for each neighbor, a counter to store the number of times that neighbor goes in TEST mode.

7.4.3 Communication Overhead

The communication overhead for our countermeasures arises from the increased message sizes mainly due to the addition of MACs. As discussed above, the TinySec project has shown that this overhead is quite small (3%). In the case of the PEAS protocol, we require that the PROBE and REPLY messages include a nonce. This would add two to four bytes to each message, depending on the size of the nonce used. In the case of ASCENT and CCP, we do not require any changes to the format of the messages used by the protocol.

Summary Our counter-measures introduce a small overhead on the protocols due to additional storage requirement, and the increased energy cost for the computation and inclusion of the MAC to authenticate messages. We believe that the extra cost introduced is reasonable considering the potential cost of using vulnerable topology maintenance protocols, e.g., being unable to obtain the desired connectivity or sensing coverage, or even worse, having a region controlled by the adversary without being aware of it.

8 Conclusions

In this paper, we have analyzed the security vulnerabilities of topology maintenance protocols for wireless sensor networks. The main contributions of the paper are

- We describe two new attacks in the context of topology maintenance protocol, namely the sleep deprivation attack and the network substitution attack. Moreover, we describe how the snooze attack can be used to reduce the sensing coverage.

- We describe how these attacks can be launched against PEAS, ASCENT, and CCP, three well-known protocols for sensor networks. Although not discussed in this paper, protocols such as GAF, CEC, AFECA, and SPAN are also vulnerable to these attacks.
- We presented counter-measures that are needed to make the protocols robust against these attacks. Authentication mechanisms can be used to prevent outsider attacks and certain insider attacks such as impersonation attacks; however, protocol-specific measures are necessary to increase the resilience of each protocol against the attacks described.

Our analysis of the security of topology maintenance protocols highlights the need for key management protocols that are resilient to node cloning and replication attacks. Finally, we show that efficient mechanisms for local (one-hop) broadcast authentication are also desirable.

References

[1] *MICA Sensor Node, http://www.xbow.com*.

[2] A. Cerpa and D. Estrin. *ASCENT: Adaptive Self-Configuring sEnsor Networks Topologies*. ACM Wireless Networks Journal, vol. 8, no. 5, Sept. 2002.

[3] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. *Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks.* ACM Wireless Networks Journal, vol. 8, no. 5, Sept. 2002.

[4] R. Di Pietro, L. V. Mancini, and A. Mei. Energy efficient node-to-node authentication and communication confidentiality in wireless sensor networks. *ACM Wireless Networks Journal, to appear. Available as Technical Report TR-04-05 at www.dsi.uniroma1.it/ dipietro/TR-04-05.pdf*.

[5] F.Ye, G. Zhong, S. Lu, and L. Zhang. *PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks.* Proceedings of the 23rd IEEE International Conference on Distributed Computing System (ICDCS'03), Rhode Island, Providence, May 2003.

[6] C. Karlof, N. Sastry, and D. Wagner. *TinySec: A Link Layer Security Architecture for Wireless Sensor Networks.* Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04), Los Angeles, California, Nov. 2004.

[7] C. Karlof and D. Wagner. *Secure Routing in Wireless Sensor Networks: Attacks and Countermeasure.* Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA'03), Anchorage, Alaska, May 2003.

[8] L. Lamport. *Password authentication with insecure communication communication.* Communications of the ACM, 24(11):770-772, Washington D.C., Nov. 1981.

[9] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar. *Spins: Security Protocols for Sensor Networks.* Proceedings of the 7th ACM International Conference on Mobile Computing and Networking (MobiCom'01), Rome, Italy, July 2001.

[10] F. Stajano and R. Anderson. *The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks.* Springer-Verlag London, UK., Cambridge, UK, Apr. 1999.

[11] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. *Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks.* Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys'03), Los Angeles, California, Nov. 2003.

[12] Y. Xu, J. Heidemann, and D. Estrin. *Adaptive Energy-Conserving Routing for Multihop Ad Hoc Networks.* Research Report527, USC/Information Sciences Institute, Oct. 2000.

[13] Y. Xu, J. Heidemann, and D. Estrin. *Geography-informed energy conservation for ad hoc routing.* Proceedings of the 7th ACM International Conference on Mobile Computing and Networking (MobiCom'01), Rome, Italy, July 2001.

[14] Y. Xu, J. Heidemann, and D. Estrin. *Energy conservation by adaptive clustering for ad-hoc networks.* Poster Session of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02), Lausanne, Switzerland, June 2002.

[15] S. Zhu, S. Setia, and S. Jajodia. *LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks.* Proceedings of the 10th ACM International Conference on Computer and Communications Security (CCS '03), Washington D.C., Oct. 2003.

Appendix

The list below summarizes the notation used in this paper.

A : network deployment area
B : subset region of A, B$\subseteq$ A
N : number of nodes of the whole network
R_T : transmission range of a node
R_S : sensing range of a sensor node
R_P : transmission range of a probe message
T_w : working time of a node running PEAS, i.e. how long it has been working
T_s : sleeping time for a node running PEAS, CCP or ASCENT
T_p : timer for the passive state in ASCENT
T_t : timer for the test state in ASCENT
K_{uv} : symmetric key shared between node u and v
K_{u_j} : j-th key of the one-way key chain of node u
MAC(k,s) : message authentication code (MAC) of message s using a symmetric key k
K_S-Coverage : an area is K_S-covered if any point P of the area is sensed by at least K_S sensors
t : value of resilience to node compromise within radio range R_T
DL : The DATA LOSS RATE measured by a node in the ASCENT protocol
LT : The loss threshold defined in ASCENT. The application data loss should not be above LT
NT : The neighbor threshold defined in ASCENT. A node can enter the TEST or ACTIVE state only if it has less than NT working neighbors.