Sec-TMP: A Secure Topology Maintenance Protocol for Event Delivery Enforcement in WSN*

Andrea Gabrielli[1], Mauro Conti[2], Roberto Di Pietro[3], and Luigi V. Mancini[1]

[1] Dipartimento di Informatica, Università di Roma "La Sapienza", Roma, IT
[2] Department of Computer Science, Vrije Universiteit Amsterdam, Amsterdam, NL
[3] Dipartimento di Matematica, Università di Roma "Tre", Roma, IT

Abstract. Topology Maintenance in Wireless Sensor Networks (WSNs), that is, alternating duty cycles with sleep cycles while having an adequate number of nodes monitoring the environment, is a necessary requirement to allow the WSNs to move from niche applications to widespread adoption; topology maintenance is even mandatory when the WSNs are used in a security sensitive context.

In this work, we present the first scalable Secure Topology Maintenance Protocol (Sec-TMP) for Wireless Sensor Networks that does not require pair-wise node confidentiality. The aim of Sec-TMP is to enforce event delivery to the BS while providing a standard topology maintenance service to the WSN. Sec-TMP enjoys the following features: it does not require pair-wise node confidentiality; it does not need any underlying routing—just one-hop communications are used; and, it is highly scalable. Sec-TMP reaches its goal being also resilient to the known attacks on TMPs: snooze attack; sleep deprivation attack; and, network substitution attack. Furthermore, Sec-TMP confines node replication attack: once a node is captured, the protocol limits the possible usage of the corresponding node's ID to a single neighbourhood. Finally, extensive simulations support our findings.

Keywords: Sensor Network Security, Topology Maintenance Protocol, Attack-Resilient.

1 Introduction

Wireless Sensor Networks (WSNs) are designed to fulfil a variety of tasks; law enforcement, disaster recovery, search-and-rescue, to cite a few [1]. WSNs are often unattended and operate in harsh environment. Furthermore, they operate without relying on existing infrastructure; for example, nodes are scattered by an airplane and once on the ground they start communicating to each other. The communication radius of a node determines its neighbourhood.

One of the most challenging research problem of WSNs is topology maintenance [6,5]: if more than the required number of nodes are present in a given

* This work is partially supported by Caspur under grant HPC-2007.

Y. Chen et al. (Eds.): SecureComm 2009, LNICST 19, pp. 265–284, 2009.

area, some of the nodes could switch from working state to sleeping state to save energy and to avoid communication congestion. Nodes in sleeping state could be activated in a further moment, if required. Different Topology Maintenance Protocols (TMPs), that assume a trusted environment, have been already proposed in the literature [6,5,13,23]. While being efficient and effective, if a malicious node is inserted in the above cited solutions, the WSN functionalities can be subverted. Indeed, just few preliminary works consider the security of TMPs, such as [18,15].

The main goal of our Sec-TMP protocol is to enforce the delivery of events intended to be received by the BS, or to detect that such a delivery failed, while providing the functionalities of a standard topology maintenance protocol. Assume a specific event (e.g. a fire alarm) is sensed in a given neighbourhood. The adversary goal could be to prevent the BS from learning such an event. To this aim, it could exploit a non-secure TMP so that only malicious nodes will be working in that neighbourhood when the BS will come to collect data—nodes that will not signal the firing event.

As an example, we can think to any data gathering scenario in a hostile environment such as a battlefield.

We assume WSNs do not necessary have an underlying routing protocol: each node is just programmed to sense data and pass-it on to the Base Station (BS). In order to increase the network lifetime, and to be resilient to attacks, nodes run our Sec-TMP protocol. In particular, assume node b is driven —by our TMP protocol— to a sleeping state while b has some data to communicate to the BS. Before setting itself in sleeping state, b sends its sensed data to its neighbour, say a, that has committed itself to be in working state (see Figure 1). Node a will eventually deliver b's data to the BS.

Contributions. In this paper we propose, to the best of our knowledge, the first Secure Topology Maintenance Protocol (Sec-TMP) for Wireless Sensor Networks (WSN) that: (i) does not require any pair-wise node confidentiality; (ii) is scalable (newly deployed nodes would be involved in the topology maintenance protocol by pre-existing nodes in the network); (iii) it is resilient to standard attacks TMP are subject to [18,15]: snooze attack; sleep deprivation attack; and, network substitution attack; and, (iv) tames the effect of node replication attack.

Organization. In Section 2, we describe the related work in the area. In sections 3 and 4 we describe the system assumptions and the threat model, respectively. In Section 5, we give an overview of the proposed protocol. In Section 6, we describe the Sec-TMP protocol in detail. In Section 7, we analyze the security of

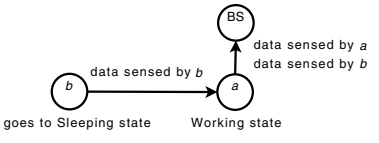

Fig. 1. Collaboration instance

our proposal and we discuss the obtained simulation results in Section 8. Finally, we conclude in Section 9.

2 Related Work

One of the main issue in WSNs is energy consumption since power is often provided to a node by a small battery, and it is often impossible to replace nodes battery. For instance, this is the case when WSNs are deployed in a hostile or not easy reachable environment.

Many different approaches have been proposed in the literature to prolong the lifetime of a WSN. A classification of these techniques is presented in [2]. There are protocols, such as [24], that are based either on changing and adjusting the transmission power of each node, or on geometrical structure-based methods to select next-hop neighbours. There is a class of so called Power Management protocols [2], such as [20], that aim to save the nodes' energy by switching off the radio in the active nodes when they do not need to communicate.

Between the proposed solutions to extend network lifetime, there are the Topology Maintenance Protocols (TMPs). The TMPs protocols leverage the node redundancy to schedule working periods between nodes. Only a subset of the nodes are in a working state, while the others goes in a sleeping state (stand-by) to save energy. Several TMPs have been proposed in the literature (e.g. SPAN [6], ASCENT [5], GAF [26], PEAS [13], CCP [23], AFECA [25]). These TMPs differ not only in the approach to schedule sleeping periods, but they differ also in their objectives. For example, SPAN [6] and ASCENT [5] aim to maintain only network connectivity. Others, such as PEAS [13] and CCP [23], aim to maintain both connectivity and sensing coverage. Some of them, as for example GAF [26], rely upon nodes location information; they require nodes with a GPS or some other location system. The CCP approach is similar to the one of SPAN, and they share the same state activity diagram and the same communication pattern.

The secure TMP protocol that we propose in this paper belongs to the Topology Maintenance Protocols that leverages sleep-wake period management. So, the comparison of our solution and the focus of this paper is within this category. In particular, all the cited protocols in the considered classification are vulnerable to attacks, as described in [15].

The first work that addressed the security issues on TMPs, [18], described the snooze attack against some previous protocols (e.g. GAF [26], SPAN [6], and AFECA [25]). However, [18] does not discuss the use of the snooze attack to reduce the sensing coverage. Moreover, [18] does not take the sleep deprivation and the network substitution attacks into consideration, nor do they discuss any possible countermeasure. In [21] the sleep deprivation attack is introduced in a context different than that of TMPs (and no countermeasures are described).

In [15], the security vulnerabilities of topology maintenance protocols for wireless sensor networks are analyzed. In particular, two new attacks in the context of topology maintenance protocol, namely the sleep deprivation attack and the

network substitution attack are described. In [15], authors describe how these attacks can be launched against PEAS, ASCENT, and CCP, and suggest some countermeasures to make these cited protocols robust against the exposed attacks.

We observe that the solutions proposed in [15] require authenticated node pair-wise communication, e.g. a pair-wise scheme such as [12,10,7] must be used in conjunction with the TMP protocol. Furthermore, [15] requires the used pair-wise key establishment scheme to be resilient to node replication attacks [8]. In particular, in the proposed countermeasures it is required that, even if the adversary captures a node w, the identity of the compromised node cannot be successfully impersonated outside the neighbourhood of w. One possible protocol for achieving this goal in a sensor network is LEAP [27]. However, any scheme that meets these requirements can be used.

In [14], the authors propose the idea of applying mechanism inspired by biological systems and processes in order to increase the security and fault-tolerance of TMP protocols. However, no practical implementation is analysed.

The motivation of the importance of events detection and data survival in unattended wireless sensor networks has recently been highlighted in [11].

Finally, also the mobility of the BS, or the mobility of the nodes, has been considered for different purposes. As an example, in [3] the mobility of the BS is leveraged to balance the power consumption of static nodes in the network. In [16], the authors proposed a protocol to move nodes along the deployment area to maintain the coverage.

3 System Assumptions and Notation

In the remainder of this work we assume the system model described in this section. In particular, we assume a static network—each node has an initial location that does not change as the time goes by. However, nodes do not need to be deployed all at the same time—newly deployed nodes will cooperate to the aim of the topology maintenance with the nodes already present in the network area.

Our Secure Topology Maintenance Protocol (Sec-TMP) does not require any underlying routing protocol—it only resorts to one hop messages. Furthermore, each node, say a, can only contact the Base Station (BS) directly (via one-hop), when the BS is within the a's transmission radius. The BS is not always reachable by every nodes. The BS has a GPS, and it moves over the network in an unpredictable way. It can also be absent from the network for a while (i.e. no node is able to reach the BS). As a result, from the single node point of view, the BS can be reachable (appear/disappear) in an unpredictable way.

In Sec-TMP, time synchronization between nodes is not necessary—nodes are not required to share a common time. However, we assume that the clock drift (difference between nodes' clock speed) is negligible.

Sec-TMP does not require any pair-wise key to be shared among nodes. Node a only shares a symmetric key, K_a, with the BS. So, we remark that the BS (and

not the other nodes) is the only one that checks the authenticity of the messages originated (not just forwarded) by the nodes. Furthermore, the nodes do not know the set of the legitimate node IDs. To ease exposition, we assume that the following hypothesis holds: for each neighbourhood, the number of nodes physically present is greater or equal than the number of nodes, (d), desired to be in a working state. Otherwise, the security properties stated in this paper cannot be guaranteed.

We define the neighbourhood of node a as the circular area having (i) center corresponding to a's location; and, (ii) radius R_t—the Transmission range—. The nodes in the neighbourhood of a are its one-hop neighbours (also called just neighbours).

Table 1 summarizes the notation used in the paper.

Table 1. Summary of Notation

Symbol	Meaning
N	Number of network nodes
d	Desired number of node in Working state for each neighbourhood
R_t	Transmission range (radius)
T_s	Sleeping time
K_a	Symmetric key shared between node a and the BS
λ	The desired probing rate towards nodes in Working state
ρ	Average of the received neighbours densities
τ	Time interval between consecutive BS contact
p	Probability that a node starts in Working state
$Dens_a$	Neighbour density of the node a
t_{BS}	Time information used by the BS
$counter_b$	A counter used by the node b

4 Threat Model

In this section, we describe the behavior and the capabilities of the adversary. In particular, we assume the adversary can compromise nodes and make them colluding, or playing an even harmful attack: having laptop-class devices storing and using information retrieved from compromised nodes. We observe that this type of attacks are particularly challenging. In fact, they cannot be prevented by authentication mechanisms since the adversary knows all the crypto material possessed by the compromised nodes.

The aim of the adversary is to avoid a sensed data to reach the Base Station. Under our assumption of data forwarding, the adversary can directly reach its target if it is able to do the following: leverage the proposed TMP protocol to avoid non-compromised (honest) nodes from being in Working state. However, the adversary could try to achieve its goal also by using standard TMP attacks [15,18]:

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

– Snooze Attack. The adversary forces the nodes to remain in sleeping state. The adversary can launch this attack to reduce the sensing coverage in a region of the network. This kind of attack can be applied to the whole network or to a subset of nodes.
– Network Substitution Attack. The adversary takes control of the entire network or of a portion of it by using a set of colluding malicious nodes.

As an example, if the adversary is able to let all the node exhaust their batteries through sleep deprivation attack, there would be no sensed data at all —so, the adversary reaches its goal.

We assume that, for a neighbourhood composed of d nodes, the adversary is able to uses at most $d-1$ compromised nodes identity. We stress that it is not necessary for the adversary to capture the node IDs in the same neighbourhood where they are intended to be used by the adversary. Furthermore, we assume that the adversary can eavesdrop on the communications of other nodes and inject data packets into the network. Nodes are not considered to be tamper-proof. As a consequence, all the information (including cryptographic keys) stored in a node the adversary compromised with are considered leaked to the adversary. As for the BS, we assume that it is trusted.

Finally, we assume the adversary is also able to perform the following type of attack (that Sec-TMP is implicitly able to defend from): Node replication attack. That is, the adversary cloning the identity (and the cryptographic material) of a captured node in other malicious nodes.

5 Protocol Overview

In Sec-TMP, each node has three operating states: Working, Sleeping, and Probing. The state diagram is described in Figure 2.

– Sleeping: the node does nothing but saving energy (i.e. turning off the radio) and waiting for the time out T_s—indicating the sleeping time—to expire. When T_s expires the node moves in Probing state.
– Probing: the node probes its neighbours to determine whether to go either in Sleeping or in Working state. The node sends a PROBE message within its transmission range R_t, and it waits for P-REPLY messages in response.
– Working: the node executes the regular node operations such as sensing and communicating to the BS as required by our protocol. When the BS claims its presence, and the node off-loads the data it stores to the BS, it also sends

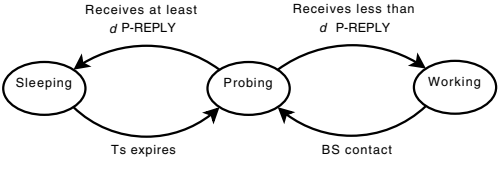

Fig. 2. State transition diagram

a request to the BS to receive a proof of the upload activity. Then, it goes in Probing state. Moreover, if the node receives a PROBE message from one of its neighbours, it replies to the neighbour sending a P-REPLY. Via the P-REPLY, the node informs the neighbours that it is in Working state.

Once a node is deployed, it starts in Sleeping state having an initial sleeping time-out T_s. The initial T_s is randomly picked over a given time interval. This choice is motivated to avoid (probabilistically) that all the nodes go in Probing state all together.

We remind that we do not assume any confidentiality layer in the node pairwise communications. The main idea underlying our proposal is to have a mechanism that, if a node a states to be in a Working state (e.g. replying to a PROBE originated by node b), then node a is actually forced to be in Working state until the next b's probe. Otherwise, a will be considered failed or malicious. In the proposed protocol implementation, we actually extend this mechanism from a single node, to a set of at least d nodes: i.e. a node a can actually move from the Working state to the Sleeping state if there is some other node that, in turn, moved from the Sleeping state to the Working state.

Enforcement is realized using the evidences issued by the BS to the node in Working state. Let us assume node b sends a PROBE to node a, and a replies being in Working state. If the BS enters the communication range of a before the next b's probe, a has to prove to b it was in Working state when the BS come —that is, it uploaded b's data to the BS. To this aim, we require a to ask the BS for a specific proof (provided by an authenticated token[1]). Node a sends back the token to b, allowing b to check that a was in Working state and interacted with the BS. This is required for any node to which a replied to as a consequence of a PROBE. However, it is not necessary that the BS releases a new token before the next b's probe. If this is the case, node a replies to b using the last token received from the BS. As we describe in Section 7.1, this behaviour does not introduce a security issue. Note that there is a specific transient case to deal with: before the first BS arrival, no node can provide a token to a probing node. We describe this situation (protocol Start-up), in Section 6.1.

6 Protocol Description

In this section, we give a detailed description of our protocol. In particular, we first present the protocol Start-up in Section 6.1. Then, we describe the behaviour of a node while it is in the different protocol states (Figure 2): Section 6.2 describes the Probing state, Section 6.3 describes the Working state and, finally, Section 6.4 describes the Sleeping state.

6.1 Protocol Start-Up

Before the first BS arrival, no node can provide a token to a probing node. As a consequence, every node could be in Working state after the first PROBE,

[1] Note that the token authenticity is enforced using symmetric keys.

until the first BS arrival. Note that this solution would be energy-consuming and could cause transmission congestion.

Assuming no adversary is present in the network before the first BS arrival, we can easily start-up our protocol as follows: instead of considering d tokens, a probing node b considers d P-REPLYs without requiring any token. In particular, the probing node b will accept P-REPLYs without requiring tokens, until it receives a P-REPLY with a token (i.e. the BS has contacted a neighbour of b). As soon as the BS arrives the first time, every nodes in Working will receive the correct tokens.

However, we do assume that an adversary can be present from the time of the network deployment (also, be there during the start-up). So, one might think to the following attack. If a single adversary node is present in every neighbourhood (this is not against our assumptions), it can send as many probe replies (P-REPLY) as it wants. In this way, a single adversary node can put every other node in the neighbourhood in Sleeping state.

To avoid this attack, we use a probabilistic approach. At the beginning, every node decides, with probability p, to go in Working state. Otherwise, with probability $1-p$, the node goes in Probing state and accepts P-REPLYs without requiring tokens. These latter nodes accept P-REPLYs without requiring tokens, until they receive a P-REPLY with a token.

As a result, even in the presence of an adversary, a honest node will be in Working state in a neighbourhood with a given probability p. Then, at the first BS arrival, it will receive the token from the BS.

Note that node a ends the start-up phase when it receives a token (either from the BS, if a is in Working state, or from a node in Working state, if a sends a PROBE). Thus, the end of the start-up propagates in the network thanks to the nodes in Working state.

In the following, we describe the operations performed by nodes when the start-up phase has come to end.

6.2 Probing State

In this section, we describe the operations that a node performs while it is in Probing state (Algorithm 1).

The aim of the Probing state is to determine the next state (Sleeping or Working) of the executing node. To do so, a node, say b, broadcasts a PROBE (line 5). The PROBE is authenticated by the node b with its key K_b, and it contains a counter value, $counter_b$. The $counter_b$ is different for each following PROBE message. As explained in Section 7.1, the $counter_b$ is necessary to avoid replay attacks. Once the PROBE has been broadcasted (line 5), the nodes waits for a time δ (set in line 4) to receive the associated reply P-REPLY from its neighbours in Working state.

In each P-REPLY message, there is a token generated by the BS. The token contains: a time information t_{BS}, the identity of the replying node a, and the neighbour density of the replying node, that is $Dens_a$.

```
input :
        b, ID of the executing node
        a, ID of the replying node
        D_reply, Desired number of P-REPLY (i.e. d)
output:
        the next node state (Working/Sleeping)
        T_s, Sleeping time
begin
    C_reply = 0
    // counter of P-REPLY
    Set TimeOut δ
    Broadcast ⟨PROBE, ID_b, counter_b, MAC_{K_b}(counter_b)⟩
    while δ not elapsed do
        for any received P-REPLY message MSG do
            if ⟨P − REPLY, b, a, t_BS, Dens_a, RMAC⟩ ← MSG AND
            MAC_{K_b}(t_BS, Dens_a, a) = RMAC then
                C_reply ++
    if C_reply < D_reply then
        move to Working state
    else
        T_s = sleepTime
        move to Sleeping state
end
```

Algorithm 1. Sec-TMP: Probing

The t_{BS} is used to inform the probing node about the time when the token has been created. This value is used in the following way: if b receives a P-REPLY with a given BS time, say t'_{BS}, then b will not consider the tokens having $t''_{BS} < t'_{BS}$ for the computation of d. In fact, let us assume node c stated to be in Working state; if it is not able to show to b a BS token with time t'_{BS}, that means c did not have the chance to communicate with the BS during the BS arrival time t'_{BS}: c was sleeping or not following the protocol. In either case, the P-REPLY of c will be ignored. Observe that the BS needs time to move and talk with the different nodes in the neighborhood. As a result, the t_{BS} values released to two neighbour nodes can actually be different by a small amount of time. For easy of exposition, in the following we do not explicitly consider this problem. However, we note that this could be solved either (i) implementing t_{BS} as just a counter managed by the BS, so not changed while talking with different nodes in the same BS passage; (ii) comparing the different t_{BS} values taking into consideration this small amount of time as negligible.

Once a node collected all the P-REPLYs, it has to take a decision about its next state. In particular, if less than the desired P-REPLYs are received (check line 10), the executing node will move to Working state (line 11). Otherwise, if the number of P-REPLYs is enough for the node, it will set the Sleeping time (line 13) and it will go in Sleeping state (line 14).

The node sets the Sleeping time according to an exponential distribution [13]:

$$f(T_s) = (\lambda/\rho)e^{-(\lambda/\rho)T_s} \tag{1}$$

where λ is the desired probing rate towards working nodes (λ is the same for each node), and ρ is the estimated neighbours density. The idea behind this is to leverage the network density; we decrease the probing rate of a sleeping node together with the increase of its neighbours density. The ρ value is computed by

each node independently as the average of the received densities ($Dens_a$ of line 8). The $Dens_a$ is the neighbour density of the replying node a. Note that the value $Dens_a$ is computed by the BS. In particular, the BS computes $Dens_a$ by counting the number of TOKEN requested by a node a in Working state. Then, the BS inserts $Dens_a$, into the TOKENs replied to a (Algorithm 2, line 6). This value represents the neighbours density of node a —node a requests a TOKEN for each of its neighbours.

6.3 Working State

In this section we describe the operations performed by a node that is contacted by the BS. When node a is aware of the presence of the BS within its communication range, it runs Algorithm 2. In particular, node a sends to the BS a request (REQ type message) containing the last PROBEs a received from its neighbours (line 2). The REQ message is authenticated using K_a. If the BS fails authenticate the message, then the BS just discards the request. However, if the authentication succeeds, with this message node a implicitly requires the BS to produce a token (that is different for each neighbour of a) to prove that a was in Working state and interacted with the BS. In fact, for each received token (TOKEN message, line 6) that passes the authentication check (line 7), a updates the token available for the corresponding neighbour b (line 8). That is, this token will be used as a P-REPLY for the next PROBE of node b. Furthermore, at the end of every BS contact, a moves to Probing state (line 10). This latter operation is required to recover from a situation that tends to put in Working state all the nodes in a neighbourhood. The latter situation is rooted in failure of a working node. Assume that d nodes in the neighbourhood are in Working state, as required by the protocol. Assume that one of these nodes, say a, exhausts its battery. As a results, less that d nodes are in Working state. In this setting, as soon as the first node in this neighbourhood, say b, sends out a PROBE, it will receive less than d P-REPLYs with a token. As a consequence, node b will go in Working state. Any other further nodes in this neighbourhood, say c, executing a probe, would receive less than d P-REPLYs with a token (also if $\geq d$ node are now in Working state), because less than d nodes would be able to send a P-REPLY with a token. So, if a working node fails, the result is that every node executing a probe in the neighbourhood of the failed node would go in Working state, having $D > d$ nodes in Working state.

However, when the BS arrives, all the nodes in Working state are required to: ask tokens for their neighbours, and go in Probing state. As a result, only the last d out of the D nodes executing the probe will go back in Working state. In general, this procedure could be leveraged to select the d out of the D nodes in some optimal way. As an example, it could be desirable that the d nodes out of D are the ones with more available energy —to do so, we can distribute the time of the probe execution of the D nodes such that the higher is the battery power, the later is the performed the probe.

Note that, before releasing a TOKEN for a node b, the BS verifies the b's PROBE message. The BS discards the request without releasing the TOKEN,

if either the authentication of the PROBE using K_b fails, or the BS has already seen the $counter_b$ value. The $counter_b$ value avoids replay attacks, as described later in Section 7.1.

Finally, we remind that the BS has a GPS: this allows the BS to build a map of the network topology as follows. The first time it receives a message from a node a, the BS bounds the location of a into a region compliant with the BS location at the moment of the contact (depending on the transmission radius of nodes). In particular, after the first contact with the BS, node a location is bounded into a circular area of radius R_t. For each following contact with node a, the BS refines the boundary region. The BS stores the network topology map for security reasons described later in Section 7.1.

```
input :
        a,  ID of the executing node
        NP_a,  PROBEs of all the neighbours node
output:
        CT_a,  Executing node's table of tokens
begin
    ⟨REQ, a, NPa, MAC_Ka(NPa)⟩ → BS
    Set TimeOut δ
    while δ not elapsed do
        for any received TOKEN message MSG do
            ⟨TOKEN, a, Dens_a, b, t_BS, MAC_Kb(t_BS, Dens_a, a), RMAC⟩ ← MSG
            if MAC_Ka(MAC_Kb(t_BS, Dens_a, a)) = RMAC then
                Update CT_a(b, ⟨TOKEN, Dens_a, t_BS, MAC_Kb(t_BS, Dens_a, a)⟩
        if received a valid TOKEN message then
            move to Probing state
end
```

Algorithm 2. Sec-TMP: BS-contact

6.4 Sleeping State

A node in Sleeping state does nothing but waiting for the T_s timer expiration. In this state, the node saves energy having its radio turned off. When the sleeping timer T_s expires, the node turns the radio on and enters the Probing state.

7 Security Analysis

We remind that the protocol goal is to enforce the delivery of a sensed event from the generator node to the BS, while providing a standard topology management service. That is, assuming at most $d - 1$ compromised nodes in the neighbourhood, there is at least a non compromised node in Working state for each neighbourhood. The idea is that if there is a sensed event that should be reported to the Base Station, there will be at least one (honest) node doing that.

As outlined in Section 4, the adversary could try to reach its goal in a direct way —i.e. leveraging the specific behaviour of our protocol— or through standard attacks on TMPs. In Section 7.1, we describe why our protocol features cannot be leveraged by the adversary. In Section 7.2, we discuss the resilience of our protocol to the standard attacks on TMPs [15,18] (to the best of our knowledge,

all the known attacks on TMPs): Sleep Deprivation attack; snooze attack; and, network substitution attack. Finally, in Section 7.3 we discuss how Sec-TMP tames the node replication attack.

7.1 Sec-TMP Security Property

In the following we first revise the possible attack, and later explain how Sec-TMP thwarts it.

Using a node's ID in more than one neighbourhood. Once an adversary captured a node, it can know the ID (and the symmetric key) of that node (we remind from Section 3 that we do not assume to have tamper proof sensors). The adversary can plug all the known IDs in a single device (e.g. a laptop class node) that pretends to be in every neighbourhood. In this way, compromising a total of d nodes, one might think that the adversary could be able to use the d IDs in every neighbourhood. So, taking over the network.

In Section 6.3 we described how the BS can estimate the network topology. We now observe that this allows the BS to prevent the described attack. In fact, assume the adversary already used a node's ID, say a, in a given location, say loc_a, to ask for tokens. Then, the BS will link the node's ID a to the claimed location loc_a. When a further asks other tokens, BS will check whether the current a's location is coherent with loc_a. If this is not the case, the BS will not give any token to a —and possibly take further actions. Furthermore, the neighbours declared by node a (for which it ask the BS appropriate tokens) should also be coherent with the other information the BS collected about the network. As an example, a node b cannot be a's neighbour if b appeared in a location not coherent with the neighbourhood area of a.

Using the same TOKENs to reply to PROBE messages with multiple identities (Sybil Attack [19]). An adversary can eavesdrop the communications between honest nodes and the BS. Assume that the honest node a requests a TOKEN for one of its neighbours, say b. The adversary eavesdrops and stores this TOKEN received by node a. Assume that the number of working neighbours of b is less than d. Then, after the following probe, node b will go in Working state. One might think that the adversary could send multiple P-REPLYs to b, using the stored TOKENs and identities different from a (Sybil attack [19]). In this way, the adversary could induce b to figure out it has more than d working neighbours —that is, the goals of the adversary is to force node b to go in Sleeping state.

However, the BS includes into each TOKEN released, the identity of the requesting node, in this case a. Note that, the TOKEN message is authenticated with the symmetric key K_b (Algorithm 2, line 6). As a consequence, the authentication check of the TOKEN (received from the adversary) performed by probing node b fails, because the sender identity does not match (check done in line 8, Algorithm 1). Thus, the attack fails because the honest node b discards the P-REPLYs sent by the adversary.

Using old BS's TOKENs to reply to PROBE messages (Replay Attack). As for the previous attack, assume that an adversary eavesdrops the communication

between honest nodes and the BS. Assume that a honest node a requests a TOKEN for one its neighbours, say b. The adversary eavesdrops and stores the TOKENs received by node a. After node a fails (for instance, because its battery depleted), the adversary could send the stored TOKENs to node b. In this way, the adversary tries to induce b to believe that a is still alive and in Working state.

Such an attack would fail for the following reason. The BS includes into the released TOKENs, a time t_{BS} (and the identity of the TOKEN requesting node, say a). The t_{BS} value changes (increases) for each BS arrival. Because of the mechanism described in 6.3, every node that ends the sleeping time before the next BS arrival will remain in Working state. Since there are at most $d-1$ compromised nodes in the neighbourhood, after a while a honest node, say c, will be in Working state (remind that, by the protocol, d nodes must be in Working state). The next time the BS will come over the neighbourhood, c will ask the BS for a TOKEN for node b. Being c a honest node, it will take the correct TOKEN with the updated t'_{BS}. When b executes the probe, if it receives a TOKEN with the updated t'_{BS}, b will ignore any other TOKEN with $t''_{BS} < t'_{BS}$.

Using old PROBE messages (Replay Attack). Assume that an adversary replies old PROBE messages of a node b to its neighbour a that is in Working state. The scope of such an attack can be to induce the BS to consider b alive even if b failed. As a result, the BS would compute a false neighbours density. Indeed, the number of TOKENs requested from a node a is used by the BS to compute the neighbours density of node a. The density estimation is then included into the TOKEN that the BS sends back to the node a. Eventually, the density is used by the neighbour of a that are in Sleeping state to set the next sleeping timer (see Section 6.2). If the adversary could successfully use old b's PROBE messages, it could maliciously falsify the BS estimation. In this way, it could increase the length of the nodes sleeping periods. Furthermore, the adversary could increase the nodes energy depletion. In fact, the BS continues to release the TOKEN for the already failed node, b, to the node a. As a consequence, node a receives the TOKEN, stores it, and then sends it to b in the following P-REPLY.

We observe that the adversary cannot succeed because of the $counter_b$ included into each PROBE. In particular, each time a node probes the neighbourhood, it includes into the PROBE message the value of the node $counter_b$. If the attacker uses old $counter_b$ values, the node a discards the corresponding PROBE requests—and possibly take further actions. Observe that, if the attacker pretends to use forged $counter_b$ values, the node a cannot further identify the $counter_b$ as a forged one (a does not know the keys that b shares with the BS). However, this can be done by the BS that know the key that b should use to generate the MAC part of the b's message.

Node impersonation. The PROBE and the REQ messages are authenticated by the senders through keyed MAC (line 5 Algorithm 1, and line 2 Algorithm 2, respectively). Thus, an adversary can send valid PROBE and REQ messages on

behalf of an identity a if, and only if, the adversary has compromised the node a (i.e. insider adversary).

Maintaining the network in start-up. Assume the adversary is present from the moment of the initial network deployment. As described in Section 6.1, the adversary can send P-REPLYs without any token, using d different identities, to maintain a node in Sleeping state. However, the adversary cannot force the nodes that start in Working state to go in Sleeping state. In fact, to do so, the adversary would need P-REPLYs with a token that can be only obtained from the BS. A node a ends the start-up when it receives a token (either from the BS, if a is in Working state, or from a node in Working state, if a sends a PROBE). Thus, the end of the start-up propagates in the network thanks to the nodes in Working state. When the node a ended the start-up, it pretends P-REPLYs with a token. As a consequence, the adversary can increase the time needed by the network to end the protocol start-up (i.e. all the nodes have received at least a P-REPLY with a token). However, it cannot indefinitely keep the network in the start-up. In Section 8.2, we report the simulation results of the time needed by the network to complete the start-up, assuming an adversary is present from the moment of the network deployment.

Event hiding. Assume the start-up completed and an adversary wants to hide to the BS an event generated by node b. To do so, the adversary could leverage the TMP protocol to have: (i) b in Sleeping state; and, (ii) just malicious nodes in Working state in the b's neighborhood—a malicious node would not send the target event to the BS.

As for (i), if the adversary wants b to move to Sleeping state, it must be able to provide d P-REPLYs with token. In fact, node b ended the start-up and it goes in Sleeping state only if it receives at least d P-REPLYs with token. The same condition is required for (ii). In fact, any other honest node should be forced to switch to Sleeping state by the adversary. Otherwise, an honest node in Working state will report the event to the BS.

As we assume that the adversary uses at most $d-1$ identity in a neighborhood, it would not be able to provide d malicious P-REPLYs.

7.2 Sec-TMP Resilience to Standard TMPs Attacks

In this section, we describe how Sec-TMP faces standard TMPs known attacks [18,15].

Resilience to Sleep Deprivation Attack. In this attack, the adversary wants to induce a node, say a, to remain in Working state, even if the node a already has d neighbours in Working state. Note that the node a, regardless if it is in Sleeping or Working state, periodically goes in Probing state. In Probing state, a sends out a PROBE. If it receives at least d P-REPLYs, it goes in Sleeping state. Thus, avoiding the reception of the P-REPLYs by node a is the only way the adversary has to successfully launch a Sleep Deprivation Attack against a. In other words, the adversary, to reach its goal, has to jam the node a to prevent

the reception of the P-REPLY messages. However, due to the periodically and asynchronously transitions of the nodes to Probing state, this operation must be performed quite often, making the attack not affordable for node-class adversary, and resource consuming even for a laptop-class adversary. Moreover, note that a denial-of-service attack involving continuous jamming (e.g. constant or deceptive jamming) can be performed in any sensor network, regardless of the topology maintenance protocol being used. Hence, we do not consider this as an attack that is specific to topology maintenance protocols. It is worth noticing that the selective jamming can also be applied during BS tokens releasing phase. However, this behaviour can be detect by the BS, that can possibly react.

Resilience to Snooze Attack. In the snooze attack, the adversary wants to induce a node, say a, to remain in Sleeping state, even if the node a has less than d neighbours in Working state, as required by the Sec-TMP protocol. As previously described in 7.1, the adversary has to compromise d node identities that are neighbours of a. Thus, Sec-TMP is resilient to an adversary that compromises up to $d-1$ nodes within a neighbourhood.

Resilience to Network Substitution Attack. The adversary substitutes legitimate nodes with malicious nodes in a portion of the network. To apply this attack, the adversary has to induce all the legitimate nodes in that portion of the network to go in Sleeping state. Thus, the resilience of Sec-TMP to this attack is the same of the resilience to the snooze attack (Section 7.2).

7.3 Sec-TMP to Thwart Node Replication Attack

We observe that our Sec-TMP protocol, while designed as a TMP for event delivery enforcement, it has also some ability to detect the node replication attack [4]: the adversary captures a node, and clone the identity (and the cryptographic material) of the captured node in other malicious nodes. In fact, as described in Section 6.3, the BS estimates the network topology. This allows the BS to detect if the same node ID, say a is used in two different locations, e.g. loc_a and loc'_a. Remind that the BS can estimate the location of (i) the node that directly asks for tokens —a in Figure 1; and, (ii) the nodes tokens are asked for —b in Figure 1. Once detected a cloned ID, the BS can take the appropriate actions, such as revoking the node.

8 Simulations and Discussion

In this section, we describe the simulation results we obtained for the Sec-TMP protocol.

We implemented a simulator of our protocol. We assumed nodes uniformly distributed in a 50x50m^2 area (nodes remain stationary after deployment). We considered deployments with N=250, 500, 750, 1000, 2000, and 4000 nodes. In particular, for each network size, the shown results are the average of 100 different random network deployment.

Table 2. Sensor parameters for simulations

Parameter	Value
R_t	10 meters
R_s	10 meters
Tx consumption	0.0074mW per bit
Rx consumption	0.003575mW per bit
Idle consumption	13.8mW per second
Sleeping consumption	0.075mW per second
Signature consumption	2% of the packet transmission cost
Initial energy of a node	60 Jules

The parameters that represent the node characteristics are reported in Table 2. The values are similar to the hardware characteristics of the Berkeley Motes [9] sensors. In particular, we use the energy model proposed in [22], and the Tiny-Sec [17] model for the power consumption of symmetric cryptography operations.

In Section 8.1, we describe the simulation results related to the network coverage lifetime. Finally, in Section 8.2, we study the start-up completion time while considering the presence of an adversary.

8.1 Network Lifetime and Area Coverage

In this subsection, we evaluate the ability of Sec-TMP to increase the coverage lifetime of the network together with the increase of the number of deployed nodes.

To measure the coverage, we logically divide the entire sensing region into adjacent 5x5m^2 patches. The coverage of the deployment area is approximated by monitoring the coverage of the top left corner of each patch, excluding those points that are on the border of the deployment area. A similar approach is used in [23]. The coverage lifetime is defined as the time interval from the activation of the network until the time of the following event: the percentage of the total area being monitored (by at least K nodes in Working state, K-coverage) drops below a specified threshold. The coverage lifetime characterizes how long the system ensures that the events are monitored with a probability of success higher than the specified threshold. In particular, we considered the threshold to be 80%. The coverage degree K is set to 1. With the minimum number of sensors in the network being equal to 250, we are quite sure we have enough nodes for 1-coverage in the area.

Figure 3 reports the coverage lifetime (y-axis) for networks with 250, 500, 750, 1000, 2000, and 4000 nodes (x-axis). Results are reported for different values of d (1,5, and 10), and for different values of BS arrival interval, τ (600 and 900 seconds). From Figure 3, we can conclude that Sec-TMP achieves the main goal of a TMP, that is the network lifetime grows almost linearly with the number of nodes. Note that, the curves trend are similar. However, while d increases, the network lifetime gain is smaller. This is because of the greater number of simultaneously active nodes. As discussed in the Section 7, the value d is related to the resilience of Sec-TMP to the adversaries. More precisely, an adversary has to compromise at least d nodes within the transmission range of a to successfully attack node a. We can say, the greater the required resilience to adversaries,

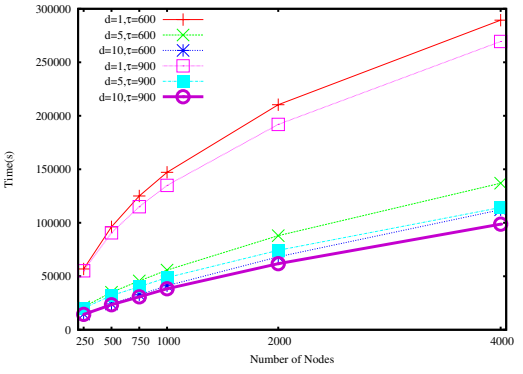

Fig. 3. Network coverage lifetime

the smaller the performance of Sec-TMP. For simulations with τ equal to 600 seconds, the gain in network lifetime is bigger than for simulations with τ equal to 900 seconds. This is expected because, as described in Section 6.3, the more nodes fails, the more nodes go in Working state; however, the number of nodes in Working state start decreasing with the BS arrival. Thus, when $\tau = 600$ the number of nodes in Working state is reduced quickly by the BS, compared to the case of $\tau = 900$. As a consequence, the increase in the network lifetime is higher with smaller τ.

8.2 Start-Up Completion Time

In this section, we study the time needed by the network to end the protocol start-up when an outsider adversary is present. In particular, we assume the adversary is present from the moment of the initial network deployment. As described in Section 6.1, the adversary can send P-REPLYs without any token to maintain a node in Sleeping state. However, the adversary cannot force the nodes that start in Working state to go in Sleeping state. In fact, to do so, the adversary would need P-REPLYs with token that can be obtained from the BS only. Without loss of generality, for the following simulations, we assume a single adversary node is present in every neighbourhood: the adversary responds to each PROBE with d P-REPLYs, using d different identities. The probability p that a node starts in Working state is equal to 0.1.

In Figure 4, we plot the network start-up time (y-axis), that is the time from the deployment of the network until all the nodes completed the start-up. Note that, a node a ends the start-up when it receives a token (either from the BS, if a is in Working state, or from a node in Working state, if a sends a PROBE). Thus, the end of the start-up propagates in the network thanks to the nodes in Working state. In Figure 4, we can see that the time to end the start-up decreases with the increase of the network density. This is due to the fact that, the higher is the network density, the higher is the probability to have a neighbour in Working state. In fact, both the number of nodes that start in Working state and the

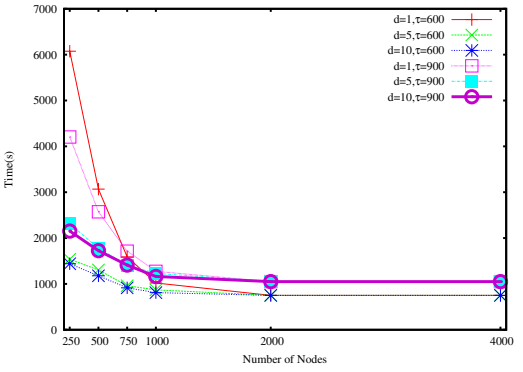

Fig. 4. Time to end the protocol start-up, when an adversary is present

number of neighbours is higher with a higher nodes density. The number of nodes that are simultaneously in Working state grows together with values of d. This is the reason why, when τ is fixed (i.e. τ=600), the time to end the start-up is higher for higher values of d. Finally, we can see from Figure 4 that the time to end the start-up increases tighter with τ. Indeed, if it takes longer for the BS to join the WSN, it takes longer to release the tokens.

9 Concluding Remarks

In this paper, we presented the first Secure Topology Maintenance Protocol (Sec-TMP) for Wireless Sensor Networks that enjoys the following features: it does not require any pair-wise node confidentiality; it does not need any underlying routing—just one-hop communications are used; and, it is highly scalable. The aim of Sec-TMP is to enforce event delivery to the BS while providing the functionalities of a standard topology management protocol. Sec-TMP reaches its goal being also resilient to the known attacks on TMPs: snooze attack; sleep deprivation attack; and, network substitution attack. Furthermore, Sec-TMP confines node replication attack: once a node is compromised, the protocol limits the possible usage of the corresponding node's ID to a single neighbourhood. Finally, simulation results support our findings.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. International Journal of Computer and Telecommunications Networking - Elsevier 38(4), 393–422 (2002)
2. Anastasi, G., Conti, M., Francesco, M.D., Passarella, A.: Mobile Ad Hoc and Pervasive Communications. In: How to Prolong the Lifetime of Wireless Sensor Networks, ch. 6. American Scientific Publishers (2007)

3. Basagni, S., Carosi, A., Melachrinoudis, E., Petrioli, C., Wang, Z.M.: Controlled sink mobility for prolonging wireless sensor networks lifetime. ACM/Springer Journal on Wireless Networks (WINET) 14(6), 831–858 (2008)
4. Bryan, P., Perrig, A., Gligor, V.: Distributed detection of node replication attacks in sensor networks. In: Proc. of the 26th IEEE International Symposium on Security and Privacy (S&P 2005), pp. 49–63 (2005)
5. Cerpa, A., Estrin, D.: Ascent: Adaptive self-configuring sensor networks topologies. IEEE Transactions on Mobile Computing 3(3), 272–285 (2004)
6. Chen, B., Jamieson, K., Balakrishnan, H., Morris, R.: Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. ACM Wireless Networks Journal 8(5), 481–494 (2002)
7. Conti, M., Di Pietro, R., Mancini, L.V.: Ecce: Enhanced cooperative channel establishment for secure pair-wise communication in wireless sensor networks. Ad Hoc Networks (Elsevier) 5(1), 49–62 (2007)
8. Conti, M., Di Pietro, R., Mancini, L.V., Mei, A.: A randomized, efficient, and distributed protocol for the detection of node replication attacks in wireless sensor networks. In: Proc. of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2007), pp. 80–89 (2007)
9. Crossbow Technology Inc. MICA Sensor Node, `http://www.xbow.com`
10. Di Pietro, R., Mancini, L.V., Mei, A.: Energy efficient node-to-node authentication and communication confidentiality in wireless sensor networks. Wireless Sensor Networks 12(6), 709–721 (2006)
11. Di Pietro, R., Mancini, L.V., Soriente, C., Spognardi, A., Tsudik, G.: Catch me (if you can): Data survival in unattended sensor networks. In: Proc. of the 6th IEEE International Conference on Pervasive Computing and Communications (PERCOM 2008), pp. 185–194 (2008)
12. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. In: Proc. of the 9th ACM International Conference on Computer and Communications Security (CCS 2002), pp. 41–47 (2002)
13. Ye, F., Zhong, G., Lu, S., Zhang, L.: Peas: A robust energy conserving protocol for long-lived sensor networks. In: Proc. of the 23rd IEEE International Conference on Distributed Computing System (ICDCS 2003), pp. 28–37 (2003)
14. Gabrielli, A., Mancini, L.V.: Bio-inspired topology maintenance protocols for secure wireless sensor networks. In: Liò, P., Yoneki, E., Crowcroft, J., Verma, D.C. (eds.) BIOWIRE 2007. LNCS, vol. 5151, pp. 399–410. Springer, Heidelberg (2008)
15. Gabrielli, A., Mancini, L.V., Setia, S., Jajodia, S.: Securing topology maintenance protocols for sensor networks: Attacks and countermeasures. In: Proc. of the 1st IEEE/CreateNet International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm 2005), pp. 101–112 (2005)
16. Jiang, Z., Wu, J., Agah, A., Lu, B.: Topology control for secured coverage in wireless sensor networks. In: Proc. of the 4th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS 2007), pp. 1–6 (2007)
17. Karlof, C., Sastry, N., Wagner, D.: Tinysec: A link layer security architecture for wireless sensor networks. In: SenSys 2004, pp. 162–175 (2004)
18. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: attacks and countermeasures. Ad Hoc Networks 1(2-3), 293–315 (2003)
19. Newsome, J., Shi, E., Song, D., Perrig, A.: The sybil attack in sensor networks: Analysis & defenses. In: Proc. of the 3rd IEEE and ACM International Symposium on Information Processing in Sensor Networks (IPSN 2004), pp. 259–268 (2004)

20. Rhee, I., Warrier, A., Aia, M., Min, J., Sichitiu, M.L.: Z-mac: a hybrid mac for wireless sensor networks. IEEE/ACM Transactions on Networking 16(3), 511–524 (2008)
21. Stajano, F., Anderson, R.: The resurrecting duckling: Security issues for ad-hoc wireless networks. In: Proc. of the 7th International Workshop on Security Protocols, pp. 172–182 (1999)
22. Wander, A., Gura, N., Eberle, H., Gupta, V., Shantz, S.C.: Energy analysis of public-key cryptography for wireless sensor networks. In: PERCOM 2005, pp. 324–328 (2005)
23. Wang, X., Xing, G., Zhang, Y., Lu, C., Pless, R., Gill, C.: Integrated coverage and connectivity configuration in wireless sensor networks. In: Proc. of the 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys 2003), pp. 28–39 (2003)
24. Wang, Y., Li, F., Dahlberg, T.A.: Energy-efficient topology control for three-dimensional sensor networks. International Journal of Sensor Networks 4(1/2), 68–78 (2008)
25. Xu, Y., Heidemann, J., Estrin, D.: Adaptive Energy-Conserving Routing for Multihop Ad Hoc Networks. Research Report 527, USC/Information Sciences Institute (2000)
26. Xu, Y., Heidemann, J., Estrin, D.: Geography-informed energy conservation for ad hoc routing. In: Proc. of the 7th ACM International Conference on Mobile Computing and Networking (MobiCom 2001), pp. 70–84 (2001)
27. Zhu, S., Setia, S., Jajodia, S.: Leap: Efficient security mechanisms for large-scale distributed sensor networks. In: Proc. of the 10th ACM International Conference on Computer and Communications Security (CCS 2003), pp. 62–72 (2003)