LHA-SP: Secure Protocols for Hierarchical Wireless Sensor Networks

Leonardo B. Oliveira, Hao Chi Wong , Antonio A. Loureiro
Computer Science Department,Federal University of Minas Gerais–Brazil
$\{$*leob,hcwong,loureiro*$\}$*@dcc.ufmg.br*

Abstract

Wireless sensor networks (WSNs) are ad hoc networks comprised mainly of small sensor nodes with limited resources, and can be used to monitor areas of interest. In this paper, we propose a solution for securing heterogeneous hierarchical WSNs with an arbitrary number of levels. Our solution relies exclusively on symmetric key schemes, is highly distributed, and takes into account node interaction patterns that are specific to clustered WSNs.

Keywords

security management, wireless sensor network security, hierarchical sensor networks

1. Introduction

Wireless sensor networks (WSNs) are ad hoc networks comprised mainly of small sensor nodes with limited resources and one or more base stations (BSs), which are much more powerful laptop-class nodes that connect the sensor nodes to the rest of the world. They are used for monitoring purposes, providing information about the area being monitored to the rest of the system. Application areas range from battlefield reconnaissance and emergency rescue operations to surveillance and environmental protection.

Like any wireless ad hoc network, WSNs are vulnerable to attacks [1, 2]. Besides the well-known vulnerabilities due to wireless communication and ad hocness, WSNs face additional problems. For instance, sensor nodes are small, cheap devices that are unlikely to be made tamper-resistant or tamper-proof. Also, they are often deployed in unprotected, or even hostile areas, which makes them more vulnerable to attacks. It is therefore crucial to add security to these networks, specially those that are part of mission-critical applications.

WSNs may be organized in a variety of different ways, and a solution designed for a flat network will unlikely be optimal for a clustered network. (In Section 2, we briefly survey different sensor network organizations.) To be effective and efficient, a solution needs to be tailored to the particular network organization at hand.

In this paper, we consider *heterogeneous hierarchical* sensor networks with *arbitrary number of levels*, and devise secure protocols for their setup, operation, and maintenance. We chose to target this class of networks because it has been shown [3] that, when compared to flat networks, they present a number of advantages including increased system throughput and decreased system delay, and increased energy savings as the number of hierarchy levels in the network is increased.

Our solution considers networks consisting largely of highly resource-constrained nodes, and uses exclusively symmetric key mechanisms. Lightweight group key based mechanisms are used whenever possible; more expensive, BS-mediated schemes are used whenever necessary. Our solution prevents intruders from taking part in network activities, tampering with or injecting messages into the network, as well as eavesdropping on communication between legitimate nodes. It is highly distributed and takes into account node interaction patterns specific to clustered WSNs. To our knowledge, this is the first work focusing on securing heterogeneous hierarchical WSNs with arbitrary number of levels.

This paper is organized as follows. In Section 2, we briefly survey existing organizations for hierarchical WSNs, and discuss their vulnerabilities and needed security. In Section 3, we present our network model. In Section 4, we present our solution. We evaluate our solution from security point of view in Section 5, and performance point of view in Section 6. Finally, we discuss related work in Section 7, and conclude in Section 8.

2. Hierarchical WSNs: Organization and Security

WSNs may be organized in different ways. In *flat* WSNs [4], all nodes play similar roles in sensing, data processing, and routing. In *hierarchical* WSNs [5], on the other hand, the network is typically organized into clusters, with ordinary cluster members and the cluster heads (CHs) playing different roles. While ordinary cluster members are responsible for sensing, the CHs are responsible for additional tasks such as collecting and processing the sensing data from their cluster members, and forwarding the results towards the BS.

Hierarchical networks can differ among themselves in various ways. They can be *homogeneous*, when all nodes except the BSs have comparable capabilities; or *heterogeneous*, when some nodes (typically the CHs) are more powerful than others. In two-level networks, CHs are found in the top level, and their children (those that belong to the cluster headed by a CH) in the lower level. In H-level networks ($H > 2$), there is a hierarchy of H nested levels, where CHs in one level are themselves children of nodes that are one level up [6]. CHs can be randomly chosen among the ordinary nodes of a homogeneous network (as in LEACH [7]), or they can be more powerful nodes that compose a heterogeneous network [8]. Clustering can also differ from one network to another. For example, under a k-hop clustering [9], members of a cluster are all within k-hops of each other. Alternatively, a subset of nodes can probabilistically self-select CHs, and the remaining nodes cluster around the CH that is geographically the closest [7].

Like any WSN, hierarchical WSNs are vulnerable to a number of attacks [1, 2] including jamming, spoofing, and replay. In these networks, attacks involving CHs are particularly damaging, because CHs are responsible for critical functions such as data aggregation and routing. If an adversary manages to become a CH, it can stage attacks such as sinkhole [1] and selective forwarding [10], thus disrupting potentially large fractions of the network.

Adversaries may leave the routing alone, and try to inject bogus sensor data into the network. Or they may choose to simply eavesdrop on communication between legitimate nodes, obtaining information that is being gathered by the BSs.

At a high level, the main security goals in a hierarchical WSN with data aggregation are: 1) access control, i.e., allow only legitimate nodes to take part in the network (e.g., become CHs and join a cluster); 2) guarantee the authenticity, confidentiality, integrity and freshness of data being passed from one member of the network to another; 3) enable data aggregation at intermediate points as sensor reports are sent from a sensor node to a BS; and 4) guarantee availability (minimize the impact of attempts of DoS attacks). In this work, we design our solution to meet these goals.

3. Our Model

We assume heterogeneous networks with two broad classes of nodes. The first class consists of a large number of highly resource-constrained sensing nodes. The second class consists of a smaller number of non-sensing nodes with various levels of resources (e.g., CPU, transmission range, and energy) responsible for data aggregation and routing.

Each node is statically assigned a hierarchy level prior to deployment (based, e.g., on its resource level), with ordinary sensor nodes being assigned level 1. We assume that nodes are deployed with some care, in such a way that level-h nodes always have level-$(h + 1)$ nodes within their communication range. Assuming that level-$(h + 1)$ nodes are always more powerful than level-h nodes, if a level-$(h + 1)$ node A is within a level-h node B's radio range, then B is within A's range.

We use the hierarchy level for clustering. Nodes in one level seek to cluster around the nodes in the next level up in such a way that the network has, at the end of the clustering process, nested clusters where level-h nodes are CHs for level-$(h - 1)$ nodes and children of level-$(h + 1)$ nodes.

Communication can then be single hop within a cluster, with the children of a cluster communicating directly with its CH. Communication with the BS is multi-hop: a message goes from a node to its CH successively until it reaches the BS. The BS can, however, communicate directly with any member of the network.

A node does not move once deployed, but can become unavailable (e.g., by energy exhaustion). When this happens, its children will try to join another cluster.

This network organization is rather static: a node's hierarchy/resource level determines whether it will be a CH, and the clustering structure formed at the initial setup of the network does not change unless a CH becomes unavailable. Nonetheless, we believe it is a reasonable starting point for investigating security in heterogeneous hierarchical WSNs.

We assume clock-driven networks: sensing reports are sent to one's CH at regular intervals. At each CH, the reports are aggregated, and only the result is passed up. Nodes have local clocks to keep track of elapsed time, for the purposes of evaluating freshness of keys and timing out on certain events. Local clocks do not need to be synchronized.

Attacks to WSNs may come from *outsiders* (those that are not legitimate members of the network) or *insiders* (those that are legitimate members of the network). The solution we propose here is meant to protect the network from attacks by outsiders only. In our model, keys can be compromised through cryptanalysis or node tampering. We assume that an attacker using either approach will succeed only after a non-negligible amount of time t, and the network can be considered secure for t units of time after deployment. We assume that BSs are trusted.

4. LHA-SP: A Suite of Secure Protocols for Hierarchical WSNs

In this section we present LHA-SP, a suite of **s**ecure **p**rotocols for **h**ierarchical **a**d hoc WSNs as modeled in Section 3. Our goal is to address the problems discussed in Section 2 (access control; authenticity, confidentiality, integrity and freshness of communications; and availability). We first give an overview of our solution (Section 4.1), then the protocol details (Section 4.2), and finally the protocol implementation (Section 4.3) .

4.1 Overview

One of the first concerns in setting up a WSN is to allow only legitimate nodes to participate in the network. To implement this access control, various cryptographic solutions (e.g., [11, 12, 13]) have been proposed. None of them is optimized for the type of networks we consider, mainly because of their key distribution schemes. In our model, each node interacts with a restricted set of nodes during the initial setup. Level-h nodes interact only with level-($h-1$) and level-($h+1$) nodes, and once the clusters are formed, this set is further reduced: a node interacts only with its CH and children. In addition, after the initial configuration, the set of nodes that a given node interacts with will change only when a CH dies and its children seek new CHs. Thus we need keys that allow legitimate nodes to recognize those that are one level up and one level down, as well as keys to protect their communications with their CH and children. Next, we first show why existing key distribution schemes do not adequately solve our problem, and then sketch our solution.

Given that public key mechanisms are inapplicable to WSNs (because of sensor nodes' resource constraints), most existing solutions rely on mechanisms that predistributes symmetric keys. There are basically three general approaches to predistributing the keys: 1) pairwise key sharing between the BS and each of the remaining nodes (e.g., [11]); 2) pairwise key sharing between ordinary nodes, which can be complete (e.g., [14]) or random (e.g., [15]); and 3) a global group keying (e.g., [16]).

In the first approach, the BS works as the key distribution center (KDC). This is rather costly in terms of communication, given that all nodes need to contact the BS to obtain keys they need to share with their CHs and children. In addition, the BS is a bottleneck.

In the second approach, two nodes that share a key at deployment have a secure link between them; those that do not, can use these links to set up their own secure links. This approach is completely distributed, and does not suffer from high communication costs or from having a bottleneck. However, to give a key to each CH-child link, each node would need to be preloaded with a large number of keys (most of them unnecessary), which is quite wasteful in the type of networks we assume.

In the third approach, everyone in the network shares the same key. This is the best in terms of cost. Each node only stores one key, and no additional keys need to be generated or exchanged. However, because the network depends exclusively on one single key, when a node is compromised, all links in the network are compromised.

In this work, we use a hybrid approach. Prior to deployment, each node is preloaded with two keys: a network-wide group key and a pairwise key it shares with the BS only. We use the group key for setting up the network. Using this key, nodes in the network organize themselves into clusters and exchange pairwise keys for securing the links between a node and its CH, needed for later network operation. Once the network is set up, the group key is discarded. The other key – shared between the node and the BS – will be used for orphan adoption, which we explain later.

The pairwise CH-Children keys enable hop-by-hop authentication, allowing data aggregation at the CHs. They also increase the network's resilience against attacks, (avoiding a wholesale compromise of the network if a node ever gets compromised).

Sometimes a network needs additional nodes. We handle addition of nodes the way we handle the initial setup, but using a new group key, which is preloaded to the new nodes, as well as propagated to all level-($h+1$) nodes (h is the hierarchy level of the new nodes).

When a node becomes orphan, the only trust association between it and the network is the key it shares with the BS. We use this key to get an orphan node back in the network. This key will not only allow the orphan to join a new cluster, but also obtain a shared key between it and its new CH. Note that even though the rejoining process depends on the BS, we assume that only few nodes will become orphans each time, and there will not be resource contention at the BS.

Notation
In the protocol specifications below, we use single capital letters (e.g., A, B) to denote network nodes; calligraphic capital letters (e.g., $\mathcal{G}$) to denote groups of nodes; $|$ to denote concatenation, and $\{m\}_K$ to denote "encryption of m using key K". $A \rightarrow B : m$ denotes "A sends a message m to B in single hop"; $A \rightarrow\rightarrow B : m$ denotes "A sends a message m to B in multiple hops"; and $A \Rightarrow \mathcal{G} : m$ denotes "A broadcasts a message m to group $\mathcal{G}$ in a single hop".

4.2 Protocol Description

In our scheme, nodes are preloaded with the following information prior to deployment: the node's id, the node's hierarchy level, a network-wide group key, a key it shares with the BS, and the current time. CHs also have information about how many keys will be used within a cluster. We explain the need for this parameter later, when we discuss security levels vs. key scopes.

Network Setup
The setup phase in LHA-SP consists of clustering and key distribution. They take place in multiple stages, in a top-down fashion. First, level-($H - 1$) nodes cluster around level-H nodes (H is the highest hierarchy level of any node in the network), and keys that will be shared pairwise between a level-($H - 1$) node and its level-H CH are generated and distributed. Then the same protocol is carried out between level-($H - 2$) and level-($H - 1$) nodes, and successively, until level-1 nodes are clustered around level-2 nodes and the keys for communication between them are set. We describe the protocol executed at each of these stages (Fig. 1) below.

At each stage, level-h nodes broadcast an adoption-ad message looking for level-($h-1$) nodes within their radio range (Step 1). This message includes the hierarchy level and the identity of the broadcasting node so that those in one level down know they are the intended recipients and who broadcast the message.

Level-($h-1$) nodes collect multiple advertisements, and use a criterion that best matches the application to choose their CHs. For example, they could choose the source of the strongest signal in a period of time. Once they choose a CH, they send an adoption-req message to the chosen node (Step 2). This message includes both the ids of the requesting node and of the chosen CH.

Upon receiving an adoption request from a node, the CH generates a symmetric key, and sends it back to the node in a send-key message (Step 3).

Note that all these message exchanges are encrypted using the group key. At each step, a node checks whether a received message originated from a legitimate network member by decrypting it using the same key. The nodes proceed with the protocol only when the check is successful.

The group key has a preset validity period (as determined by each sensor's local clock) after which it expires and is discarded by each of the nodes. Thus, the setup protocol should be completed before the key expires.

At the end of this protocol (after all $H - 1$ stages have been executed), each node will have acquired $c + 1$ pairwise keys: one shared with its CH, and the remaining c shared between it and each of its children.

Note that we do not need to use message authentication code (MAC) to guarantee message integrity, because all communications are single hop, and the messages do not go through intermediate nodes where they could potentially be corrupted maliciously. To handle non-malicious corruptions from the environment, we use mechanisms such as CRC (Cyclic Redundancy Check). Doing so, we avoid adding additional bytes to the original message, and save on energy that would be spent sending these bytes.

Adoption being broadcast by level-h nodes (e.g. A_h, B_h):

1. $A_h \Rightarrow \mathcal{G}_{h-1}$: $\{\textsf{adoption-ad} \mid h \mid id_A\}_{K_\mathcal{G}}$
   $B_h \Rightarrow \mathcal{G}_{h-1}$: $\{\textsf{adoption-ad} \mid h \mid id_B\}_{K_\mathcal{G}}$
   ...

Nodes from $\mathcal{G}_{h-1}$ (e.g., M_{h-1}, N_{h-1},O_{h-1}) choose their CHs (e.g., B_h, A_h) and respond:

2. $M_{h-1} \rightarrow B_h$: $\{\textsf{adoption-req} \mid id_M \mid id_B\}_{K_\mathcal{G}}$
   $N_{h-1} \rightarrow A_h$: $\{\textsf{adoption-req} \mid id_N \mid id_A\}_{K_\mathcal{G}}$
   $O_{h-1} \rightarrow A_h$: $\{\textsf{adoption-req} \mid id_O \mid id_A\}_{K_\mathcal{G}}$
   ...

Level-h nodes (e.g., A_h) generate and distribute pairwise keys
to be shared with each of their children (e.g. N_{h-1}, O_{h-1}):

3. $A_h \rightarrow N_{h-1}$: $\{\textsf{send-key} \mid id_A \mid id_N \mid K_{A,N}\}_{K_\mathcal{G}}$
   $A_h \rightarrow O_{h-1}$: $\{\textsf{send-key} \mid id_A \mid id_O \mid K_{A,O}\}_{K_\mathcal{G}}$
   ...

The various symbols denote:

X_h :	a node X from level h
$\mathcal{G}_h$:	the group of all nodes from level h
h :	hierarchy level
id_X :	id of node X
$K_\mathcal{G}$:	the network-wide group key
$K_{X,Y}$:	pairwise key shared between nodes X and Y

Figure 1: The setup protocol.

Network Operation

Once the network is set up and the normal operation begins, there will be two types of communications: child-CH communications, which consist mainly of sensing reports; and CH-children communications, which consist mainly of network management messages.

In child-CH communications, a child A_h simply encrypts the message m_A with the key it shares with the CH D_{h+1}. For freshness, a nonce n_A can be added before encryption.

$$A_h \rightarrow D_{h+1} : \{\textsf{sensing} \mid n_A \mid m_A\}_{K_{A,D}}$$

At each hop, the CH can decrypt messages it received, examine their content, and carry out data aggregation before sending the aggregate result forward.

Information that flows the opposite direction, i.e., from the BS to the rest of the WSN, can be destined to a particular node or a subset of the nodes. If the information is destined to a single node, our pairwise keying scheme is completely adequate. In cases where the information is destined to a larger number of nodes, it can be distributed, multi-hop, through the intermediate CHs. Note that whenever a CH needs to send the same information to multiple of its children, the best mechanism would be an authenticated broadcast, which cannot be done with our CH-children pairwise keying. However, we can use the pairwise keys to bootstrap the scheme proposed by LEAP [12], in which broadcasts are authenticated using keys in a hash key chain. Alternately, we can group all the children in a cluster in a few groups, and have each group share a key. E.g., given a cluster with 10 children, there will be 10 keys if we use pairwise keys between the CH and each of its child. There will be 5 keys if each key is shared between the CH and 2 of its children. And one key if all members of the cluster share the same key. The idea is that, when a CH needs to broadcast a message to multiple nodes in the cluster, it can make fewer transmissions: one for each group that shares a key (instead of one for each child). This scheme thus trades security (the scope of a key) with efficiency. In any case, we expect each CH to have a reasonable small number of children (according to [3], between 4% and 10% of a network must be composed of CHs for maximum energy efficiency). And given that there are typically few network management messages, it is not impractical for the CHs to deliver these messages to each child separately, encrypted with the key they share.

Network Maintenance
During the lifetime of a network, nodes come and go: existing nodes may depart from the network (e.g., by energy exhaustion) and new nodes may be added. We handle these changes as follows.

Adding New Nodes To securely add new nodes to the network, we follow the general scheme used in the initial deployment. Nodes about to be added are preloaded with the same set of data as in the initial deployment; however, the group key and the clock time will have new values. The group key is now a newly generated key (the initial one has expired), and the time is the current time given by the operator preloading these values. The new group key is intended to be the trust association between the nodes being added and those already in the network. Thus, both the new group key and the current time also need to be known by all pre-existing level-h nodes, where h is one hierarchy level higher than the level of nodes being added. The BS can transmit these values using single hop communication to the intended recipients.

1. $A_h \Rightarrow \mathcal{G}_{h+1}$: $\{\textsf{new-node-ad} \mid h\}_{K'_\mathcal{G}}$
2. $B_{h+1} \Rightarrow \mathcal{G}_h$: $\{\textsf{adoption-ad} \mid (h+1) \mid id_B\}_{K'_\mathcal{G}}$
3. $A_h \rightarrow B_{h+1}$: $\{\textsf{adoption-req} \mid id_A \mid id_B\}_{K'_\mathcal{G}}$

Level-$(h+1)$ nodes (e.g., B_{h+1}) generate and distribute keys to the new children (e.g., A_h):

4. $B_{h+1} \rightarrow A_h$: $\{\textsf{send-key} \mid id_B \mid id_A \mid K_{B,A}\}_{K'_\mathcal{G}}$

All symbols as previously defined; $K'_\mathcal{G}$ is a newly generated group key.

Figure 2: Node addition protocol.

Fig. 2 shows the node addition protocol. Unlike the initial setup protocol, here new nodes seeking to join the network advertise their intention through a new-node-ad message (Step 1), which includes the hierarchy level h of the node broadcasting the message. Those at level $h+1$ that hear this broadcast reply with adoption-ad, signaling their intention to adopt. The rest of the protocol is identical to the initial setup protocol (Fig. 1).

Just like before, the new group key expires after a predefined period of time, before which all new nodes should have joined the network.

Node A_h being adopted by node B_{h+1}

1.	$A_h \Rightarrow \mathcal{G}_{h+1}$:	orphan-ad $\mid h$
2.	$B_{h+1} \rightarrow \mathcal{G}_h$:	adoption-ad $\mid (h+1) \mid id_B$
3.	$A_h \rightarrow B_{h+1}$:	adoption-req $\mid m, \mathsf{MAC}(K_{AS}, m)$
4.	$B_{h+1} \rightarrow C_{h+2}$:	$\{$key-req $\mid m \mid n_B, \mathsf{MAC}(K_{AS}, m), \mathsf{MAC}(K_{BS}, m \mid n_B)\}_{K_{BC}}$
5.	$C_{h+2} \rightarrow\rightarrow S$:	$\{$key-req $\mid m \mid n_B, \mathsf{MAC}(K_{AS}, m), \mathsf{MAC}(K_{BS}, m \mid n_B)\}_{K_{CD}}$

BS S authenticates A and B, and generates K_{AB}

6.	$S \rightarrow A_h$:	key-del $\mid \{id_B \mid n_A \mid K_{AB}\}_{K_{AS}}$
7.	$S \rightarrow B_{h+1}$:	key-del $\mid \{id_A \mid n_B \mid K_{AB}\}_{K_{BS}}$

Symbols as previously defined, with the following additions:

$m = id_A \mid id_B \mid n_A$

n_X : nonce produced by node X

Figure 3: Orphan adoption protocol.

Orphan adoption We assume that the network provides means for children of a cluster to learn the unavailability of its CH. This can be achieved, e.g., by periodically pinging the CH, or by using mechanisms such as watchdog [10].

Whenever a CH becomes unavailable, it is desirable for the orphans to join another cluster. Given that the pairwise key shared between an orphan and the BS is the only trust association shared between the orphan and the network, we use the BS as an authentication authority and KDC. The protocol (Fig. 3) works as follows.

First, the orphan nodes broadcast the orphan-ad message searching for a new CH (Step 1). This message includes the level h of the orphan. Upon receiving an orphan-ad message, candidate CHs (i.e., those one level up) reply with adoption-ad (Step 2). Neither message is protected, given that the communicating parties do not share any keys.

Each orphan then chooses one among all those that sent a reply, and responds with adoption-req (Step 3). This message is authenticated by a message authentication code, produced with the key the orphan shares with the BS. It is not destined to the chosen CH, but will be included in the following (key-req) message the CH sends to the BS (step 4). Note that we use a message authentication code to protect the message here, because the message travels multiple hops before it is received by the intended recipient, and is thus subject to malicious corruption.

For the key-req message (Step 4), the CH adds its own message authentication code (produced using the key it shares with the BS) to the message authentication code from the orphan. It then encrypts the result using the key it shares with its own CH. The encryption offers link level security (and will be replaced at each hop), whereas the message authentication codes are intended for the BS to verify the originators of the request.

After checking the authenticity of both the orphan and the adopting CH, the BS generates a symmetric key and sends it, single hop, to both. These messages (Steps 6 and 7) do not include a MAC code because their transmissions are single hop (as justified earlier). The orphan node is now back on the network, and there is a secure communication channel between it and its CH.

4.3 Protocol Implementation

Given the resource-constraints, the protocols specified above need to have efficient implementations. Thus, cryptographic algorithms need to be chosen not only by their security strength, but also by the amount of resource they consume. In this work, we take advantage of the building blocks from SPINS [11], a suite of lightweight symmetric key based security protocols for highly resource-constrained WSNs. We briefly describe these building blocks below.

To save memory, SPINS implements all cryptographic primitives using one single block cipher; RC5 [17] was chosen because of its small code size and its efficiency. Encryption and decryption in SPINS are stream ciphers obtained from using RC5 in the counter (CTR) mode. Message authentication code (MAC) is implemented using RC5 under the CBC-MAC [18] mode: the target message is encrypted under CBC mode, and the message authentication code is the output from the last stage. The same MAC function is used to generate pseudo-random numbers (e.g., nonces) needed by the security module. MAC(K, c) produces a sequence of pseudo-random numbers if the value of c is incremented after each generation. Following good security practice, SPINS uses different keys for different cryptographic functions, all of them derived from a master key χ. The MAC function is also used for this derivation. Using different values of p in MAC(χ, p), different computationally secure keys can be derived from the master key. Thus, one can, e.g., derive different keys for encryption and MAC code. Or even different keys for different communication directions; i.e., one key for communications from A to B, and another from B to A.

We use the building blocks described above to implement our protocols. In case of encryption and decryption, a counter value is actually needed in each operation, as they are implemented by RC5 under CTR mode. Because the counter value determines the one-time pad produced by RC5, and one-time pads should not be used twice for security reasons, all encryptions produced using a given key should use different counter values.

In our proposal, counters are dealt with differently depending on the type of keys used. When pairwise keys are used, as e.g. in child-CH communications, counters are not sent between the parties. Instead, they are kept at both ends of the link, and incremented after each encryption (this is the approach used by SPINS). This is feasible because when pairwise link keys are used, the two communicating parties can keep track of counter values that have been used in conjunction with their link key. (The parties can actually get de-synchronized. But they can either try successive increments, or execute simple synchronization protocols to re-synchronize.) When group keys are used, as e.g. in the setup protocol, counters can no longer be synchronized implicitly as above, because not all nodes will hear all transmissions encrypted using the group key, and therefore, would not know which counter has or has not been used. In these cases, we append the counter value being used to each ciphertext. To prevent different nodes from using the same counter, we assign different non-overlapping ranges of values to each node in the network. Each node is expected to start with the smallest value in its range, and successively use increasing values in successive encryptions.

5. Security Analysis

5.1 Network setup

The security of our setup protocol depends on two assumptions: 1) that an adversary will take a certain amount of time to compromise the group key or tamper with a node, and 2) that this amount of time exceeds that required to set up the network.

Under these two assumptions, our protocol guarantees that only the legitimate nodes of the network can become CHs, join a cluster, distribute keys and receive them. This is because all message exchanges in the setup protocol are encrypted with the group key, which is known only by the members of the network.

The pairwise keys generated by the CHs and distributed to each of their children are encrypted by the group key before they are transmitted. Thus any network member could potentially eavesdrop on communications intended to some other node, and learn the value of a pairwise key it should not know. However, according to our assumptions, 1) legitimate members of the network would not eavesdrop (misbehave, in general), unless they have been tampered with; and 2) node tampering would take longer than the network setup time. Thus, at the end of the protocol, every node that had the group key would have assured its place in the network topology, and each link would have associated with it a pairwise key, known only by the CH that generated it and the child that is its intended recipient.

Note that it is possible for an adversary to capture all this encrypted traffic for later evaluation, after the group key is compromised. Using this approach, the adversary can obtain pairwise keys that were distributed, and use them for eavesdropping or impersonation. The scope of the compromise is limited to the region within which the initial sniffing took place.

5.2 Network Operation

During the network operation, communication between any node and its CH is secured by the pairwise key they share. This ensures confidentiality and authentication of communication between the two, prevents bogus nodes from tampering with and injecting messages, and allows data aggregation to take place at the CH. Replay of old messages is prevented by the use of nonces (which is actually dispensable, given that each new encryption is produced with a different counter value).

The group key expires right after the network setup and is not used thereafter. Thus compromise of a single node has limited scope, and would compromise only the links protected by the keys found in the compromised node.

5.3 Network Maintenance

Adding New Nodes

The node addition protocol follows quite closely the initial setup protocol. Thus, the discussion in Section 5.1 applies here. The new group key, used to bootstrap the operation, is known only to legitimate and interested parties: it is preloaded to the nodes being added, and delivered securely to the relevant CHs by the BS.

Orphan Adoption

The goal of our orphan adoption protocol is to re-insert an orphan (and the subtree rooted at it) securely into the network routing topology, and to provide it with a key to communicate with the rest of the network securely .

Our proposal relies on the BS as an authentication authority and KDC. The BS authenticates both the adoption-req message (step 3, Fig 3) from the orphan and the key-req message (step 4, Fig 3) from the new CH, before it generates and delivers the requested key. Both the requests and the key delivery are protected by the pairwise keys shared between the BS and the nodes. This means that 1) only requests from legitimate members of the network will be processed; and 2) only the orphan and its new CH will learn the value of the new key, which will be used to secure the communication between them.

Note that because the orphans do not share any trust associations (keys) with the nodes that can potentially adopt them, the messages sent in steps 1 and 2 (Fig. 3) are not protected. This is a source of vulnerability. For instance, a bogus node can send a large number of orphan-ad messages to the network, and try to trigger a response to each of its messages, with the intent of consuming the resources of some of the nodes in the network. Another possible attack is for an intruder to impersonate a potential adopter, and send an adoption-ad message (step 2) in response to orphan-ad messages. The intruder can simply quit the protocol here or try to submit a key-req message (step 3). In any case, the orphan will be left waiting for a key that will never come, and the adoption process will never be completed. We can address the first attack by limiting the number of orphan-ad messages a potential CH will handle per period of time. This is reasonable because we assume that only a small number nodes will become orphans at the same time. To handle the second attack, an orphan can set a waiting time, and if it does not hear from the BS before this time expires, it will contact another potential adopter.

6. Performance Evaluation

In this section, we consider the overhead incurred by our protocols, as compared to a stripped down version of the protocols without the security devices. For example, the stripped down version of the setup protocol would consist of steps 1 and 2 (Fig. 1) only, and the messages exchanged in these steps would not be encrypted.

Due to space constraints, we focus on the protocols for setup and network operation. In what follows, n_h denotes the total number of level-h nodes.

6.1 Communication and Computational Overhead

For the setup protocol (Fig. 1), security incurs the following cost:

- Each adoption-ad and adoption-req message transmission incurs one encryption operation and c additional bytes for counter value. adoption-ad message sends are executed once by all the nodes in the network, except those at level 1; and adoption-req sends are executed once by all the nodes, except those at the highest level.
- Each adoption-ad and adoption-req message reception incurs one decryption operation and reception of c additional bytes for counter value. Each level-h node receives no more than n_{h+1} adoption-ad messages, and the set of all level-h nodes will receive a total of n_{h-1} adoption-req messages.
- send-key messages are exchanged only in the secure version of the protocol. Each transmission incurs one key generation, one encryption, and the transmission itself. Each reception incurs one decryption and the message reception itself. The set of all level-h nodes will send a total of n_{h-1} such messages, whereas each node in the network (except those at the highest level) will receive only one such message.

Our setup protocol is quite scalable. The number of interactions between a level-h node A and level-$(h+1)$ nodes is bounded by $n_{(h+1)}$; and that between A and level-$(h-1)$ nodes is bounded by the number of children in the cluster headed by A.

In Child-CH communication, the overhead incurred by security is negligible: one encryption at the sender and one decryption at the receiver. In CH-children communication, the overhead incurred by security depends on the number of children a CH tries to reach each time. In the best case scenario, the CH has a message destined to a single child. The overhead is then simply one encryption at the CH and one decryption at the child. In the worst case, the transmission from the CH is destined to all the children in the cluster. In such scenarios, our solution is expensive. Instead of a broadcast, the CH needs to send a separate message to each of the children (respectively, group of children), because of our pairwise (respectively, group) keying scheme.

To be concrete, we estimate energy consumption incurred by our solution in a 2-level network consisting of 10 Mica2 motes [19] at level 2, and 100 Mica2dot motes [19] at level 1, deployed in such a way that each Mica2 receives 10 adoption requests, and form clusters that has 10 Mica2dots in them. (In real deployments, the clustering will likely be more random. However, the ideal deployment considered here can give us an initial picture on more realistic deployments.) This 1:10 ratio between the number of CHs and the number of ordinary nodes is within the range in which energy savings are the largest [3].

We consider the network under five different setup and operation conditions: one with no security devices, and the rest with varying degrees of security, as determined by the scope of the keys used in CH-children and communications. We use "–" to refer to the scenario with no security, and "1", "2", "5", and "10" to refer to scenarios where 1, 2, 5, and 10 keys are used within the cluster.

We use the following consumption rates [20]: 16.25 μJ/byte and 12.25 μJ/byte for Mica2 mote transmission and reception, respectively; and 15 μJ for encrypting (or decrypting) 8 bytes of data using RC5. We consider the same values for Mica2dot mote, even though they have lower consumption rates due to slower clock rate (4MHz vs 8MHz). We assume 36-byte packets (the maximum used in TinyOS [21]).

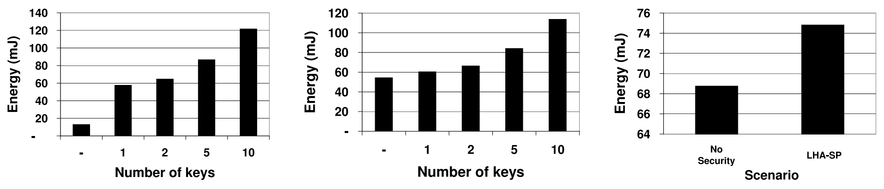
Our estimates show that the LHA-SP setup protocol consumes from 344% to 836% more energy when compared to the plain protocol (Fig. 4(a)) – note, however, that setup is performed just once. The overhead incurred by LHA-SP in CH-Children communication is high when 5 or 10 keys are used per cluster (Fig. 4(b)). CH-Children communications are used for network management functions, and do no occur frequently. Thus, this high cost is likely to be tolerable.

For Child-CH communication, which occurs the most in a WSN, LHA-SP is quite efficient: it incurs only 9% more energy. This is expected: all child-CH communications are pairwise, and can be efficiently protected by pairwise keys.

6.2 Storage Overhead

To estimate the storage overhead incurred by security in CH-children and child-CH communications, we modified the source code for Surge [22] (an application in the TinyOS distribution that periodically sends data to the BS) so that it sends and receives (RC5-based) encrypted data, instead of plaintext. We used cryptographic code from TinySec.

Compared to the original Surge, the modified version takes 990 bytes more in ROM (of which the motes have a total of 128KB) and 164 bytes more in RAM (of which the motes have a total of 4KB). Storage overhead incurred by encryption is thus negligible.



(a) Setup (b) CH-Children communication (c) Child-CH communication

Figure 4: Energy Consumption

Regarding the keys, each level-h node, $h > 1$, needs to keep the group key, two pairwise keys, and k keys shared with its children. In the worst case, when each child share a unique key with its CH, k is equal to the size of the cluster. Level-1 nodes have not children, and have to keep just three keys. The storage cost is therefore $O(k)$ for level-h nodes ($h > 1$), and $O(1)$ for level-1 nodes.

As a whole, we conclude that LHA-SP is efficient and scales gracefully in terms of computation, communication, and storage costs.

7. Related Work

The number of studies specifically targeted to security of resource-constrained WSNs has grown significantly. Due to space constraints, we provide a sample of studies based on cryptographic methods and focus on those targeted to hierarchical WSNs.

Perrig et al. [11] proposed a suite of symmetric key based security building blocks, which we use in our solution. Eschenauer et al. [15] looked at random key distribution schemes, and originated a large number of follow-on studies which we do not list here. Most of the proposed key distribution schemes, probabilistic or otherwise (e.g., [12]), are not tied to specific network organizations, although they mostly assume flat network with multi-hop communication; thus they are not well suited to hierarchical WSNs. Still others (e.g., [20]) focused on detecting and dealing with injection of bogus data into the network.

Among those specifically targeted to hierarchical WSNs, Carman et al. [14] suggested using higher powered nodes for key generation and management functions. Kong et al. [23] devised a solution for a specific hierarchical and heterogeneous network. Their proposal uses RSA certificates, and implements end-to-end security at transport layer. Bohge and Trappe [13] proposed an authentication framework for a concrete 2-tier network organization, in which a middle tier of more powerful nodes between the BS and the ordinary sensors were introduced for the purpose of carrying out authentication functions. In their solution, apart from the ones in the lowest tier, nodes carry out public key operations. More recently, Ferreira et al. [24] devised symmetric key based security extensions for LEACH [7], a WSN routing protocol which dynamically and periodically rearranges its clustering.

8. Conclusion

In this paper, we proposed a solution for securing heterogeneous hierarchical WSNs with arbitrary number of levels. Our solution provides security for network setup and reconfiguration, as well as for the normal network operation traffic. Our scheme sets up pairwise keys between a CH and each of its children (or group of children) using lightweight group key based mechanisms whenever possible, falling back on more expensive, BS-mediated mechanisms whenever necessary.

Our solution is highly distributed, takes into account node interaction patterns that are specific to clustered WSNs, and enables data aggregation at CHs.

We also evaluated the overhead incurred by our solution. The results showed that the overhead incurred by our protocols in terms of energy consumption ranges from fairly small (in cases where multiple children in a cluster share a key) to tolerable (in cases where each child share a pairwise key with its CH). We conclude that our solution is practical.

References

[1] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's AdHoc Net. Journal, Special Issue on Sensor Net. Applications & Protocols*, 1(2-3):293–315, 2003.

[2] Anthony D. Wood and John A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, October 2002.

[3] Enrique J. Duarte Melo and Mingyan Liu. The effect of organization on energy consumption in wireless sensor networks. In *IEEE Globecom 2002*, November 2002.

[4] I. F. Akyldiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38:393–422, December 2002.

[5] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Mobile Computing and Networking*, pages 263–270, Seattle, WA USA, 1999.

[6] E.M. Belding-Royer. Multi-level hierarchies for scalable ad hoc routing. *Wirel.Net.*, 9(5):461–478, 2003.

[7] W.R. Heinzelman, A.Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *IEEE Hawaii Int. Conf. on System Sciences*, pages 4–7, january 2000.

[8] V. Mhatre, D. Kofman C. Rosenberg, R. Mazumdar, and N. Shroff. A minimum cost heterogeneous sensor network with a lifetime constraint. *IEEE Transaction on Mobile Computing*, 2004.

[9] Yaacov Fernandess and Dahlia Malkhi. K-clustering in wireless ad hoc networks. In *Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 31–37. ACM Press, 2002.

[10] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 255–265, 2000.

[11] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, September 2002.

[12] S. Zhu, S. Setia, and S. Jajodia. Leap: efficient security mechanisms for large-scale distributed sensor networks. In *10th ACM conference on Computer and communication security*, pages 62–72, 2003.

[13] Mathias Bohge and Wade Trappe. An authentication framework for hierarchical ad hoc sensor networks. In *Proceedings of the 2003 ACM workshop on Wireless security*, pages 79–87. ACM Press, 2003.

[14] D. W. Carman, P. S. Kruus, and B. J. Matt. Constraints and approaches for distributed sensor network security. Technical report, NAI Labs, The Security Research Division, Network Associates, Inc., 2000.

[15] Laurent Eschenauer and Virgil D. Gligor. A key management scheme for distributed sensor networks. In *9th ACM conference on Computer and communications security*, pages 41–47, 2002.

[16] Stefano Basagni, Kris Herrin, Danilo Bruschi, and Emilia Rosti. Secure pebblenets. In *2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 156–163. ACM Press, 2001.

[17] Ronald L. Rivest. The RC5 encryption algorithm. In Bart Preneel, editor, *Fast Software Encryption*, pages 86–96. Springer, 1995. (Proceedings Second International Workshop, Dec. 1994, Leuven, Belgium).

[18] Data encryp. standard modes of operation. Federal Information Processing Standards Publication, 1981.

[19] Crossbow Technology. *MPR/MIB Mote Hardware Users Manual – Document 7430-0021-05*, Dec 2003.

[20] Fan Yea, Haiyun Luo, Songwu Lu, and Lixia Zhang. Statistical en-route filtering of injected false data in sensor networks. In *INFOCOM 2004*, 2004.

[21] J. Hill, R. Szewczyk, A. Woo, S. Hollar, David Culler, and K.Pister. System architecture directions for networked sensors. In *ACM 9th ASPLOS'03*, pages 93–104, 2000.

[22] Crossbow Technology. *Getting Started Guide Revision A – Document 7430-0022-04*, April 2004.

[23] J.Kong, H. Luo, K. Xu, D. L. Gu, M. Gerla, and Sl Lu. Adaptive Security for Multi-layer Ad-hoc Networks. *Wireless Communication and Mobile Computing*, 2(5):533–547, 2002. Special Issue.

[24] A. C. Ferreira, M. A. Vilaça, L. B. Oliveira, E. Habib, H. C. Wong, and A. A. Loureiro. On the security of cluster-based communication protocols for wireless sensor networks. In *4th International Conference on Networking (ICN 2005)*. To appear.