On the Security of Cluster-based Communication Protocols for Wireless Sensor Networks

Adrian Carlos Ferreira[1], Marcos Aurélio Vilaça[1], Leonardo B. Oliveira[1], Eduardo Habib[1], Hao Chi Wong[1], Antonio A.Loureiro[1]

Federal University of Minas Gerais, MG,Brazil
Email: {adrian, vilaca, leob, habib, hcwong, loureiro}@dcc.ufmg.br

Abstract Wireless sensor networks are ad hoc networks comprised mainly of small sensor nodes with limited resources, and are rapidly emerging as a technology for large-scale, low-cost, automated sensing and monitoring of different environments of interest. Cluster-based communication has been proposed for these networks for various reasons such as scalability and energy efficiency. In this paper, we investigate the problem of adding security to cluster-based communication protocols for homogeneous wireless sensor networks consisting of sensor nodes with severely limited resources, and propose a security solution for LEACH, a protocol where clusters are formed dynamically and periodically. Our solution uses building blocks from SPINS, a suite of highly optimized security building blocks that rely solely on symmetric-key methods; is lightweight and preserves the core of the original LEACH.

1 Introduction

Wireless sensor networks (WSNs) are ad hoc networks comprised of small sensor nodes with limited resources and one or more base stations (BSs), which are much more powerful nodes that connect the sensor nodes to the rest of the world. WSNs are used for monitoring purposes, and can be used in different application areas, ranging from battlefield reconnaissance to environmental protection.

Cluster-based communication protocols (e.g., [1]) have been proposed for ad hoc networks in general and sensor networks in particular for various reasons including scalability and energy efficiency. In cluster-based networks, nodes are organized into clusters, with cluster heads (CHs) relaying messages from ordinary nodes in the cluster to the BSs. This 2-tier network is just an example of a hierarchically organized network that, in general, can have more than two tiers.

Like any wireless ad hoc network, WSNs are vulnerable to attacks [2,3]. Besides the well-known vulnerabilities due to wireless communication and ad hocness, WSNs face additional problems, including 1) sensor nodes being small, cheap devices that are unlikely to be made tamper-resistant or tamper-proof; and 2) their being left unattended once deployed in unprotected, or even hostile areas (which makes them easily accessible to malicious parties). It is therefore crucial to add security to WSNs, specially those embedded in mission-critical applications.

Adding security to WSNs is specially challenging. Existing solutions for conventional and even other wireless ad hoc networks are not applicable here, given the lack of resources in sensor nodes. Public-key-based methods are one such example. In addition, efficient solutions can be achieved only if tailored to particular network organizations.

In this paper, we investigate the problem of adding security to cluster-based communication protocols for homogeneous WSNs (those in which all nodes in the network, except the BSs, have

comparable capabilities). To be concrete, we use LEACH (Low Energy Adaptive Clustering Hierarchy) [1] as our example of protocol. LEACH is interesting for our investigation because it rearranges the network's clustering dynamically and periodically, making it difficult for us to rely on long-lasting node-to-node trust relationships to make the protocol secure.

To the best of our knowledge, this is the first study focused on adding security to cluster-based communication protocols in homogeneous WSNs with resource-constrained sensor nodes. We propose SLEACH, the first version of LEACH with cryptographic protection, using building blocks from SPINS [4]. Our solution is lightweight and preserves both the structure and the capabilities of the original LEACH.

In what follows, we first discuss related work (Section 2), then introduce LEACH and discuss its main security vulnerabilities (Section 3). We then present SLEACH (Section 4), analyze its security and evaluate its performance (Section 5).

2 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 access control.

Perrig et al. [4] proposed a suite of efficient symmetric key based security building blocks, which we use in our solution. Eschenauer et al. [5] looked at random key predistribution 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., [6]), are not tied to particular network organizations, although they mostly assume flat network, with multi-hop communication; thus they are not well suited to clustered networks. Still others (e.g., [7,8]) focused on detecting and dealing with injection of bogus data into the network.

Among those specifically targeted to cluster-based sensor networks, Bohge et al. [9] 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, only the sensor nodes in the lowest tier do not perform public key operations. More recently, Oliveira et al. [10] propose solution that relies exclusively on symmetric key schemes and is suitable for networks with an arbitrary number of levels.

3 LEACH and its Vulnerabilities

LEACH assumes two types of network nodes: a more powerful BS and resource-scarce sensor nodes. In homogeneous networks with resource-scarce sensor nodes, nodes do not typically communicate directly with the BS for two reasons. One, these nodes typically have transmitters with limited transmission range, and are unable to reach the BS directly. Two, even if the BS is within a node's communication range, direct communication typically demands a much higher energy consumption. Thus, nodes that are farther away usually send their messages to intermediate nodes, which will then forward them to the BS in a multi-hop fashion. The problem with this approach is that, even though peripheral nodes actually save energy, the intermediate nodes, which play the role of routers, end up having a shortened lifetime, when compared with other nodes, since they spend additional energy receiving and transmitting messages.

LEACH assumes every node can directly reach a BS by transmitting with sufficiently high power. However, to save energy and avoid the aforementioned problem, LEACH uses a novel type of routing that randomly rotates routing nodes among all nodes in the network. Briefly, LEACH works in rounds, and in each round, it uses a distributed algorithm to elect CHs and dynamically cluster the remaining nodes around the CHs. To avoid energy drainage of CHs, they do not remain CHs forever; nodes take turns in being CHs, and energy consumption spent on routing is thus distributed among all nodes. Using a set of 100 randomly distributed nodes, and a BS located at 75m from the closest node, simulation results show that LEACH spends up to 8 times less energy than other protocols [1].

Protocol Description Rounds in LEACH (Fig. 1) have predetermined duration, and have a *setup* phase and a *steady state* phase. Through synchronized clocks nodes know when each round starts and ends.

The setup consists of three steps. In the *advertisement* step, nodes decide probabilistically whether or not to become a CH for the current round (based on its remaining energy and a globally known desired percentage of CHs). Those that will broadcast a message (adv) advertising this fact, at a level that can be heard by everyone in the network. To avoid collision, the CSMA-MAC protocol is used. In the *cluster joining* step, the remaining nodes pick a cluster to join based on the largest received signal strength of a adv message, and communicate their intention to join by sending a join_req (join request) message using CSMA-MAC. Given that the CHs' transmitters and receivers are calibrated, balanced and geographically distributed clusters should result. Once the CHs receive all the join requests, the *confirmation* step starts with the CHs broadcasting a confirmation message that includes a time slot schedule to be used by their cluster members for communication during the steady state phase.

Once the the clusters are set up, the network moves on to the steady state phase, where actual communication between sensor nodes and the BS takes place. Each node knows when it is its turn to transmit, according to the time slot schedule. The CHs collect messages from all their cluster members, aggregate these data, and send the result to the BS. The steady state phase lasts much longer compared to the setup phase.

Security vulnerabilities Like most of the protocols for WSNs, LEACH is vulnerable to a number of security attacks including jamming, spoofing, replay. But because it is a cluster-based protocol, relying on their CHs for routing, attacks involving CHs are the most damaging. If an intruder manages to become a CH, it can stage attacks such as sinkhole and selective forwarding, thus disrupting the network. The intruder may also leave the routing alone, and try to inject bogus sensor data into the network, one way or another. A third type of attack is passive eavesdropping.

4 Adding Security to LEACH

Attacks to WSNs may come from *outsiders* or *insiders*. In cryptographically protected networks, outsiders do not have credentials (e.g., keys or certificates) to show that they are members of the network, whereas insiders do. Insiders may not always be trustworthy, as they may have been compromised, or have stolen their credentials from some legitimate node in the network. The solution we propose here is meant to protect the network from attacks by outsiders only. Another rather ordinary trust assumption we make is that BSs are trusted.

In this section, we add two of the most critical security properties to LEACH: **data authentication** (it should be possible for a recipient of a message to authenticate its originator), and **data freshness** (it should be possible for a recipient of a message to be sure that the message is not a replay of an old message). We focus on devising a solution to prevent an intruder from becoming a CH or injecting bogus sensor data into the network by pretending to be one of its members. Our solution uses building blocks from SPINS [4], a suite of lightweight security primitives for resource-constrained WSNs.

4.1 SPINS Overview

SPINS [4] consists of two symmetric-key security building blocks optimized for highly constrained sensor networks: SNEP and μTESLA. SNEP provides confidentiality, authentication, and freshness between nodes and the BS, and μTESLA provides authenticated broadcast. μTESLA implements the asymmetry required for authenticated broadcast using one-way key chains constructed with cryptographically secure hash functions, and delayed key disclosure. μTESLA requires loose time synchronization. See [4] for further details on SPINS.

4.2 Overview of Our Solution

SLEACH needs an authenticated broadcast mechanism to allow non-CHs to authenticate the broadcaster as being a particular, legitimate, node of the network. Our small nodes do not have, however, the resource level needed to run μTESLA (it requires the sender to store a long chain of symmetric keys).

We propose a solution that divides this authenticated broadcast into two smaller steps, leveraging on the BS, which is trusted and has more resources. In a nutshell, assuming that each sensor node shares a secret symmetric key with the BS, each CH can send a slightly modified `adv` message, including the id of the CH in plaintext (which will be used by the ordinary nodes as usual) and a message authentication code (MAC[1]) generated using the key the CH shares with the BS (the MAC will be used by the BS for the purpose of authentication). Once all these (modified) `adv` messages have been sent by the CHs, the BS will compile the list of legitimate CHs, and send this list to the network using the μTESLA broadcast authentication scheme. Ordinary nodes now know which of the (modified) `advs` they received are from legitimate nodes, and can proceed with the rest of the original protocol, choosing the CH from the list broadcast by the BS.

We can modify the rest of the setup protocol similarly. However, this would require that BS to authenticate each and all nodes of the network at the beginning of each round, which is not only prohibitively expensive, but also makes BS a bottleneck of the system. Thus, we leave these messages unauthenticated, and argue, in Section 5.1, why this decision does not bring devastating consequences, as long as we add an lighter-weight corrective measure.

4.3 Protocol Details

Predeployment Each node X is preloaded with two keys: χ_X, a master symmetric key that X shares with the BS; and k_n, a group key that is shared by all members of the network. For freshness purposes, each node X also shares a counter C_X with the BS.

[1] Note that MAC is often used to stand for medium access control in networking papers. In this paper, we use MAC to stand for message authentication code.

From χ_X, the key holders derive K_X, for MAC computation and verification. k_n is the last key of a sequence S generated by applying successively a one-way hash function f to an initial key k_0 ($S = k_0, k_1, k_2, \ldots, k_{n-1}, k_n$, where $f(k_i) = k_{i+1}$). The BS keeps S secret, but shares the last element k_n with the rest of the network.

Setup Phase: Advertisement Once it decides to be a CH, a node H broadcasts a sec_adv message (step 1.1, Fig. 2), which is a concatenation of its own id with a MAC value produced using K_X. Ordinary nodes collect all these broadcasts, and record the signal strength of each. The BS receives each of these broadcasts, and verifies their authenticity.

Once the BS has processed all the sec_adv messages, it compiles the list V of authenticated H's, identifies the last key k_j in S that has not been disclosed (note that all key k_i, such that $i > j$, have been disclosed, whereas all key k_i, such that $i \leq j$, have not), and broadcasts V (step 1.2) using μTESLA and k_j. k_j is disclosed after a certain time period (step 1.3), after all nodes in the network have received the previous message.

Cluster Joining After receiving both the broadcast and the corresponding key, ordinary nodes in the network can authenticate the broadcast from the BS and learn the list of legitimate CHs for the current round. (Note that the key is authentic only if it is a an element of the key chain generated by the BS, and immediately precedes the one that was released last. That is, if $f(k_j) = k_{j+1}$.) They then choose a CH from this list using the original algorithm (based on signal strengths), and send the join_req message (step 2) to the CH they choose. Note that this message is unprotected, and identical to message 2, Fig. 1

Confirmation After the CHs receive all the join_reqs, they broadcast the time slot schedule to their cluster members (step 3). Depending on the security level required, we can take advantage of the same procedure used to authenticate sec_adv messages here.

Steady-state Phase During this phase, sensor nodes send measurements to their CHs (step 4). To authenticate the origin of these measurements, they also enclose a MAC value produced, using the key they share with the BS. The CHs aggregate the measurements, and transmit the aggregate result, authenticated, to the BS; the MACs from the cluster members, are simply forwarded in mac_array messages, as they are unable to verify them. Note that the number of mac_array messages is dependent on the size of the cluster (step 5.2).

The BS verifies both the MAC value generated by the CH, as well as the ones from the ordinary nodes. Unless all verifications are successful, the BS will discard the corresponding aggregate result, and the originators of failed MACs will be seen as intruders. In case there are intruders among the ordinary nodes, the BS will report their identities to their CHs, which will then drop message from these nodes for the remaining of the round.

Note that the MAC values from the ordinary nodes do not take the measurements into account. In fact, they should not be, as the BS would not be able to verify them (note that the BS do not learn the measurements themselves, but only their aggregate value). Thus, the MAC values, in this case, only authenticate the fact that the key holder has sent one more message (as the counter value has been incremented). The BS needs to trust that the CHs indeed used the reports from their members to generate the aggregate result.

Also note that most of the messages (except the MACs from the non-CHs) travel single hop. This means that they do not go through intermediate nodes where they could potentially be corrupted

maliciously. Thus, in this work, we use MAC just for authentication purposes. To handle non-malicious corruptions of messages from the environment, in single hop communications, we use mechanisms such as CRC – Cyclic Redundancy Check.

In this paper, for the purpose of simplicity, we assume that there are no additional control messages, aside from the ones we show. It is not difficult to see, however, that they can be handled the way setup messages are.

5 Security Analysis and Performance Evaluation

5.1 Security Analysis

In designing SLEACH, our goal was to implement access control, and prevent intruders from participating the network. We discuss below how well we achieve it.

Our solution allows authentication of sec_adv messages (steps 1.1, 1.2, and 1.3, Fig. 2), and prevents intruders from becoming CHs. Thus, unless there are insider attacks, the network is protected against selective forwarding, sinkhole, and HELLO flood attacks [2]. Note that we leave the confirmation message (step 3, Fig. 2) unauthenticated; and an intruder would be able to broadcast bogus time slot schedules, possibly causing DoS problems in the communication during the steady-state phase. We argue that the intruder will likely have simpler ways (jamming, e.g.) to accomplish the same objective.

Instead of trying to become CHs, intruders may try to join a cluster, with three goals in mind: (1) To send bogus sensor data to the CH, and introduce noise to the set of all sensor measurements; (2) To have the CH forward bogus messages to the BS, and deplete its energy reserve; and (3) To crowd the time slot schedule of a cluster, causing a DoS (Denial of Service) attack, or simply lowering the throughput of a CH. Our solution does not prevent intruders from joining the clusters (join_req messages, step 2, Fig. 2, are not authenticated), but does prevent them from achieving the first goal. Their rogue measurements (or better, the aggregate report that embed these measurements) will be discarded by the system, as they are unable to generate MACs that can be successfully verified by the BS, during the steady-state phase. Note that this verification also guarantees freshness, as the counter value should have been incremented from last time. We also prevent the intruders from achieving the second goal: their CHs will cease to forward their messages, once they are flagged and reported by the BS (again because of MACs that cannot be verified). Our scheme cannot prevent the intruders from achieving the third goal, but we argue that it can be accomplished by other much easier means, such as jamming the communications channels, for example.

Our solution does not guarantee data confidentiality. To do so, while still preserving the data fusion capability, pairwise keys shared between CHs and their cluster members would be needed.

5.2 Performance Evaluation

Our solution is extremely simple: each node, aside from the BS, is preloaded with only two keys, one for the BS to authenticate the legitimate members of the network, and the other for authenticated broadcasts from the BS.

In terms of communication and processing overhead, the SLEACH setup protocol incurs to the BS and negligible overhead: one authenticated broadcast and one key disclosure. For the CHs, sending a sec_adv message instead of adv incurs one MAC computation and energy for transmitting the MAC bits. For the non-CHs, the additional work has to do with receiving and processing the

BS's authenticated broadcast (steps 1.2 and 1.3, Fig. 2), the computation overhead consisting of one MAC and one (a few in cases where desynchronization occurs) application of f. All these overheads are minimum.

For the steady-state phase, SLEACH has all nodes send authenticated messages, which requires one MAC computation and additional MAC bits in the message. In addition, the CHs also forward MACs from their cluster members to the BS. Taking the current values (e.g., cluster size from [11], and MAC size from [4]) into account, we believe that this overhead is tolerable.

6 Conclusion

To the best of our knowledge, this is the first study focused on adding security to cluster-based communication protocols in homogeneous WSNs with resource-constrained sensor nodes. We proposed SLEACH, the first modified version of LEACH with cryptographic protection against outsider attacks. It prevents an intruder from becoming a CH or injecting bogus sensor data into the network.

SLEACH is quite efficient, and preserves the structure of the original LEACH, including its ability to carry out data fusion.

The simplicity of our solution relies on LEACH's assumption that every node can reach a BS by transmitting with sufficiently high power. Thus, we expect our solution to be applicable to any cluster-based communication protocol where this assumption holds. In cases where it does not hold, alternative schemes are needed. This is topic for future work.

References

1. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: IEEE Hawaii Int. Conf. on System Sciences. (2000) 4–7
2. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: Attacks and countermeasures. In: Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols. Volume 1. (2003) 293–315 Also appeared in First IEEE International Workshop on Sensor Network Protocols and Applications.
3. Wood, A.D., Stankovic, J.A.: Denial of service in sensor networks. IEEE Computer **35** (2002) 54–62
4. Perrig, A., Szewczyk, R., Wen, V., Culler, D., Tygar, J.D.: SPINS: Security protocols for sensor networks. Wireless Networks **8** (2002) 521–534 Also appeared in Mobile Computing and Networking.
5. Eschenauer, L., Gligor, V.D.: A key management scheme for distributed sensor networks. In: Proceedings of the 9th ACM conference on Computer and communications security, ACM Press (2002) 41–47
6. Zhu, S., Setia, S., Jajodia, S.: Leap: efficient security mechanisms for large-scale distributed sensor networks. In: Proceedings of the 10th ACM conference on Computer and communication security, ACM Press (2003) 62–72
7. Przydatek, B., Song, D., Perrig, A.: SIA: Secure information aggregation in sensor networks. In: ACM SenSys 2003. (2003) 175–192
8. Yea, F., Luo, H., Lu, S., Zhang, L.: Statistical en-route filtering of injected false data in sensor networks. In: INFOCOM 2004. (2004)
9. Bohge, M., Trappe, W.: An authentication framework for hierarchical ad hoc sensor networks. In: Proceedings of the 2003 ACM workshop on Wireless security, ACM Press (2003) 79–87
10. Oliveira, L.B., Wong, H.C., Loureiro, A.A.F.: LHA-SP: Secure protocols for hierarchical wireless sensor networks. In: 9th IFIP/IEEE International Symposium on Integrated Network Management (IM'05), Nice, France (2005) To appear.
11. Melo, E.J.D., Liu, M.: The effect of organization on energy consumption in wireless sensor networks. In: IEEE Globecom 2002. (2002)

Setup phase

1. $H \Rightarrow \mathcal{G} : h, \mathsf{adv}$
2. $A_i \rightarrow H : a_i, h, \mathsf{join\_req}$
3. $H \Rightarrow \mathcal{G} : h, (\ldots, \langle a_i, T_{a_i} \rangle, \ldots), \mathsf{sched}$

Steady-state phase

4. $A_i \rightarrow H : a_i, d_{a_i}$
5. $H \rightarrow BS : h, \mathcal{F}(\ldots, d_{a_i}, \ldots)$

Figure 1. LEACH protocol

Setup phase

1.1. $H \Rightarrow \mathcal{G} : h, \mathsf{mac}_{kh}(h \mid ch \mid \mathsf{sec\_adv})$
A_i : store(h)
BS : if $\mathsf{mac}_{kh}(h \mid ch \mid \mathsf{sec\_adv})$ is valid
add(h, V)

1.2. $BS \Rightarrow \mathcal{G} : V, \mathsf{mac}_{k_j}(V)$

1.3. $BS \Rightarrow \mathcal{G} : k_j$
A_i : if $(f(k_j) = k_{j+1})$ and $(h \in V)$)
h is authentic

2. $A_i \rightarrow H : a_i, h, \mathsf{join\_req}$
3. $H \Rightarrow \mathcal{G} : h, (\ldots, \langle a_i, T_{a_i} \rangle, \ldots), \mathsf{sched}$

Steady-state phase

4. $A_i \rightarrow H : a_i, d_{a_i}, \mathsf{mac}_{ka_i}(a_i \mid ca_i)$

5.1. $H \rightarrow BS : h, \mathcal{F}(\ldots, d_{a_i}, \ldots), \mathsf{mac}_{kh}(h \mid ch \mid \mathcal{F}(\ldots, d_{a_i}, \ldots))$

5.2. $H \rightarrow BS : h, \mathsf{mac\_array}, (\ldots, a_i, \mathsf{mac}_{ka_1}(a_i \mid ca_i), \cdots), \mathsf{mac}_{kh}(h \mid ch))$

6. $BS \rightarrow H$: intruder ids

Figure 2. SLEACH protocol

The various symbols denote:

H, A_i : A CH and an ordinary node, respectively
$\mathcal{G}$: The set of all nodes in the network
$\Rightarrow, \rightarrow$: Broadcast and unicast transmissions, respectively
a, h : Node ids
adv, join_req, sched, mac_array : String identifiers for message types
$\langle x, T_x \rangle$: A node id x and its time slot T_x in its cluster's TDMA schedule
d_x : Sensing report from node x
$\mathcal{F}$: Data fusion function
V : An array of node ids
kx : Symmetric key shared by X and BS
cx : Counter shared by node X and BS
$\mathsf{mac}_{kx}()$: MAC calculated using kx
$f()$: One-way hash function
add(x, V) : Add id x to V
store(x) : Store id x for future validation