Adding Security to Cluster-based Communication Protocols for Wireless Sensor Networks

Adrian Carlos Ferreira[1] , **Marco Aurélio Vilaça**[1] , **Hao Chi Wong**[1]

[1]Computer Science Department – UFMG
Belo Horizonte, Brazil

{adrian, vilaca, wong}@dcc.ufmg.br

Abstract. *Wireless sensor networks are ad hoc networks comprised mainly of small sensor nodes with limited resources, and can be used in pervasive computing environments to monitor areas 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 mainly of small sensor nodes with limited resources (low power, low bandwidth, and low computational and storage capabilities) and one or more base stations (BSs), which are much more powerful nodes that connect the sensor nodes to the rest of the world. In pervasive computing environments, WSNs are used for monitoring purposes, providing information about the area being monitored to the rest of the system. WSNs can be used in different application areas, ranging from battlefield reconnaissance and emergency rescue operations to surveillance and environmental protection.

Cluster-based communication protocols like [4, 7, 6, 10] has been proposed for ad hoc networks in general and sensor networks in particular for various reasons including scalability and energy efficiency. In a cluster-based communication, nodes are typically organized into clusters, with cluster heads (CHs) relaying messages from ordinary nodes in the cluster to the BSs. This 3-tier network is just an example of a *hierarchically organized network* that, in general, can have $n \geq 3$ tiers. Hierarchical networks can be *homogeneous* if all the nodes in the network (with the exception of the BSs) have comparable computational, storage, and communication power; or *heterogeneous* if nodes in different hierarchical levels have different levels of resources.

Like any wireless ad hoc network, wireless sensor networks are vulnerable to attacks [5, 11]. Besides the well-known vulnerabilities due to wireless communication and *ad hoc*ness, WSNs face additional problems. For instance, sensor nodes are small,

cheap devices that are unlikely to be made tamper-resistant or tamper-proof. In addition to their nodes' lack of physical protection, these networks are often left unattended once deployed in open, unprotected, or even hostile areas. This makes them easily accessible to random individuals and malicious parties alike. It is therefore crucial to add security to these networks, specially those that are part of 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 low-power sensor nodes. Public-key-based methods are one such example.

In this paper, we investigate the problem of adding security to cluster-based communication protocols for homogeneous WSNs. We aim for lightweight mechanisms that are consistent with the resource scarcity of sensor nodes.

To be concrete, we use LEACH (Low Energy Adaptive Clustering Hierarchy) [4] as our example communication protocol. In addition to assuming homogeneous networks with resource-constrained nodes, LEACH is an interesting protocol for our investigation because it rearranges the network's clustering dynamically and periodically. This dynamic and periodical reorganization makes it difficult for us to assume static trust relationships, which we might leverage on to make the protocol secure.

To the best of our knowledge, this is the first study that focuses on adding security to cluster-based communication protocols in homogeneous WSNs with resource-constrained sensor nodes. We propose secure-LEACH, the first secure version of LEACH, using building blocks from SPINS. Our solution is lightweight and preserves the structure of the original LEACH.

The rest of this paper is structured as follows. We first discuss related work (Section 2). Then, in Section 3, we introduce LEACH and discuss its main security vulnerabilities. In Section 4, we show how we make LEACH secure. We discuss our solution in Section 5, and conclude in Section 6.

2. Related Work

Karlof and Wagner [5] did a comprehensive study on secure routing in WSNs. They surveyed the major classes of attacks against these networks, and suggested broad class of countermeasures that can be applied in each case. They also studied a number of protocols that have been proposed in the literature (including LEACH), and pointed out their main vulnerabilities. They do not, however, offer concrete security protocols for any of the routing protocols.

Boghe and Trappe [1] proposed an authentication framework for hierarchical sensor networks. They assume, however, nodes with heterogeneous capabilities in different levels of hierarchy, some powerful enough to perform public key operations. In fact, their solution makes use of public key methods.

Perrig *et al* proposed SPINS [9], a suite of symmetric key based security building blocks which we use in our solution. Oliveira *et al* [8] also proposed a pure symmetric key based solution, but they assume hierarchical n-level (arbitrary n) networks with heterogeneous nodes.

WSNs require novel key distribution schemes, because of their resource constraints. Eschenauer and Gligor proposed the first work [3] on random key predistribution. There are variations of this basic schema [2].

Stankovic [11] surveys DOS attacks against wireless sensor networks, which we do not address in this work.

3. LEACH and its Vulnerabilities

To make our research concrete, we use LEACH as our underlying communication protocol. In this paper, for the purpose of simplicity, we assume that there are no additional control messages, aside from the ones we show.

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. This approach is inadequate (inefficient) except for few unlikely configurations, where e.g., the BS is located in the center of a circle of sensor nodes.

Another, more energy efficient, alternative would be to take advantage of intermediate nodes as routers. Nodes that are farther away 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 nodes that are not routing, 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, one hop transmission directly to a BS can be a high power operation, and is specially inefficient given the amount of redundancy typically found in WSNs. Multihop communication, on the other hand, has the aforementioned problem of energy drainage in router nodes. To solve this problem, LEACH uses a novel type of routing that randomly rotates routing nodes among all nodes in the network, thus distributing energy consumption among all network elements.

Briefly, LEACH works in rounds, and in each round, it uses a distributed algorithm to dynamically cluster the nodes. Each cluster has a single CH, who is responsible for collecting packets from its cluster members, and forward them to the BS. Due to these additional receptions and transmissions, CHs have a much higher energy consumption, compared to the other nodes. To address this issue, a CH does not remain a CH forever; nodes take turns in being CHs, and energy consumption spent on routing is thus distributed among all nodes.

Using a set of 100 nodes randomly distributed, 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 [4]. The energy saving comes from other sources other than just dynamic cluster-based communication: data fusion (CHs do data fusion before sending them to the

BS), node sleeping (given that only CHs need to forward messages, the remaining nodes are activated only when they themselves are transmitting, and remain in sleep mode a good part of the time), and transmitter calibration (nodes calibrate their transmitters in such a way that they are only high enough to reach the CH).

Protocol Description

LEACH operates in rounds with predetermined duration. Through synchronized clocks nodes know when each round starts and ends. There are two phases in each round: *setup* and *steady state*.

The setup phase starts with each node probabilistically deciding whether or not to become a CH for the current round based on its remaining energy and a globally known desired percentage of CHs. This self-election algorithm is such that all nodes have the same probability of becoming CHs during the lifetime of the network. This phase ends with non-CH nodes clustering around the CHs. The setup phase consists of the following steps:

Advertisement: Nodes decide whether or not to become a CH for the current round. Those that will broadcast a message (adv) advertising this fact. adv is broadcast at a level that can be heard by everyone in the network. To avoid adv collision the CSMA-MAC protocol is used.

Cluster Joining: Once the remaining nodes hear advs from all the CHs, they 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 request join_req using CSMA-MA. Given that the CHs' transmitters and receivers are calibrated, balanced and geographically distributed clusters should result.

Confirmation: Once the CHs receive all the join requests, they broadcast a confirmation message that includes a TDMA schedule to be used by their cluster members for communication during the steady 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 TDMA schedule), and thus can turn its transmitter and receiver off outside this period to save energy. 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 routing protocols for wireless sensor networks, LEACH is vulnerable to a number of security attacks [5], including jamming, spoofing, replay, etc. But because it is a cluster-based protocol, relying fundamentally on their CHs for routing, attacks involving CHs are the most damaging. If a intruder manages to become a CH, it can stage attacks such as sinkhole and selective forwarding, thus disrupting the network. Of course, the intruder may 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.

Setup phase

1. $H \Rightarrow \mathcal{G}$: h, adv
2. $A_i \rightarrow H$: $a_i, h, \mathsf{join\_req}$
3. $H \Rightarrow \mathcal{G}$: $h, \langle a_1, T_{a_1} \rangle, \ldots, \langle a_n, T_{a_n} \rangle, \mathsf{sched}$

Steady-state phase

1. $A_i \rightarrow H$: $message$
2. $H \rightarrow BS$: $message$

The various symbols denote

H, A_i :	A cluster head 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 A and H's ids, respectively
adv, join_req, sched :	String identifiers for message types
$\langle a, T_a \rangle$:	A node id a and its time slot T_a in its cluster's TDMA schedule

Figure 1: Leach protocol

4. Adding Security to LEACH

Attacks to WSNs may come from *outsiders* or *insiders*. In protected networks, outsiders do not have credentials (e.g., keys or certificates) to show that they are members of the network. Insiders are nodes or agents that have these credentials. Insiders may not always be trustworthy, given that they may be otherwise trustworthy nodes that have been compromised, or they may have stolen their credentials from some legitimate node of the network. The solution we propose here is meant to protect the network from attacks by outsiders only. In the rest of this paper, we use intruders to mean outside attackers. Another rather ordinary trust assumption we make is that BSs are trusted.

In this section, we will add some of the most critical security properties to LEACH:

Data authentication: It should be possible for a recipient of a message to authenticate its originator.

Data confidentiality: Confidential data can be protected in such a way that only those that are supposed to have access to them actually do.

Data integrity: It should be possible for a recipient of a message to be sure that the message was not modified while in transit.

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.

Using primitives that guarantee these properties, we focus on devising 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. We also protect sensor data from being eavesdropped. Our solution uses building blocks from SPINS [9], a suite of lightweight security primitives for resource-constrained WSNs.

4.1. SPINS Overview

SPINS 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 [9] for further details on SPINS.

4.2. Overview of Our Solution

One straightforward way to prevent intruders from infiltrating a network (and becoming CHs or injecting bogus messages) is to use a globally shared key for link layer encryption and authentication. Using this key to protect their communication, members of the network can certify that a given message they received is actually from some other legitimate node of the network. In the case of LEACH, we could encrypt all the messages in Fig. 1, and adv in particular, using such a key. Systems that use globally shared keys, however, are known to be fragile: a compromise of a single node would compromise the whole network.

To address this problem, we would need to use keys with smaller scope. In the case of the adv message (message 1, Fig. 1), we would need an authenticated broadcast mechanism, i.e., a mechanism that would allow non-CH nodes to authenticate the broadcaster as being a particular, legitimate, node of the network. Public key systems would be perfect for this purpose, but they are inapplicable here because of the amount of resource they require. Even μTESLA [9], which is a symmetric key based mechanism that implements the asymmetry required in this context is too costly, given that it requires the sender to store a long chain of symmetric keys, which requires storage capabilities not present in our small nodes.

We propose a solution that divides this authenticated broadcast into two smaller steps, leveraging on the BS, who is trusted and has more resources. In a nutshell, assuming that each sensor node shares a secret symmetric key with the BS, then each CH can send a slightly modified adv message. Part of this message would have the id of the CH in plaintext, which will be used by the ordinary nodes as usual. A second part would be protected (in reality, a MAC) using the key the CH shares with the BS, and 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, and authenticate the messages for join request and confirmation. This solution is prohibitively expensive, however, because the BS would need to authenticate each and all nodes of the network at the beginning of each round, making it a bottleneck of the system. Thus, for efficiency and scalability reasons, we leave these messages unauthenticated, and show below why this would not bring devastating consequences, as long as we add an lighter-weight corrective measure.

If we do not authenticate join_req messages (message 2, Fig. 1), intruders will be able to join any of the cluster. Intruders that join a cluster may have three goals in mind: (1) To crowd the TDMA schedule of a cluster, causing a DOS attack, or simply lowering the throughput of a CH; (2) To send bogus sensor data to the CH, and introduce noise to the set of all sensor measurements; and 3) To have the CH forward bogus messages to the BS, and deplete its energy reserve.

We chose not to try to address the first possibility, since there are simpler ways for an intruder to accomplish the same objective, by jamming the communication channels, for example. To prevent the second possibility, we authenticate and encrypt sensor data from sensor nodes to the BS using the symmetric key they share. To thwart the third possibility, we have the BS warn the CHs about the presence and the identities of intruders, as soon as it detects their presence (through messages that the BS cannot decrypt successfully).

If we leave the confirmation message (message 3, Fig. 1) unauthenticated, then an intruder would be able to broadcast bogus TDMA schedules, possibly causing DOS problems in the communication during the steady-state phase. Here too, an intruder has simpler ways (jamming, e.g.) to accomplish the same objective.

4.3. Protocol Details

We describe secure-LEACH (Fig. 2) in the rest of this section.

Notation

In what follows, we use χ_A to represent the master symmetric key that node A shares with the BS. From this key are derived other keys used in a secure communication between the two: K_A for encryption, and K'_A for MAC computation. For freshness purposes, each node A also shares a counter C_A with the BS. Something more about the counter. We use 'unicast' communication to refer to "logical" unicasts, i.e., communication that is addressed to one party only, even if the message is physically a broadcast (which all wireless radio communications are).

4.3.1. Predeployment

Each node A of the network is preloaded with two keys: χ_A, a master symmetric key that A shares with the BS; and k_n, a group key that is shared by all members of the network.

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 $f(k_i) = k_{i+1}$).

Setup phase

1. $H \Rightarrow \mathcal{G}$: $h, \mathsf{mac}_{kh'}(h|\mathsf{adv})$
   A : store(h)
   BS : if $\mathsf{mac}_{kh'}(h|\mathsf{adv})$ is valid
   add(h, V)
2. $BS \Rightarrow \mathcal{G}$: msg $= V, \mathsf{mac}_{k_j}(V)$
3. $BS \Rightarrow \mathcal{G}$: k_j
   A : if $(f(k_j) = k_{j+1})$ and $(h \in V)$)
   h is authentic
4. $A \rightarrow H$: $a, h, \mathsf{join\_req}$
5. $H \Rightarrow \mathcal{G}$: $h, \langle a_1, T_{a_1} \rangle, \ldots, \langle a_n, T_{a_n} \rangle, \mathsf{sched}$

Steady-state phase

1. $A \rightarrow H$: msg $= a, E_{ka,ca}(d), \mathsf{mac}_{ka'}(ca|E_{ka,ca}(d))$
2. $H \rightarrow BS$: msg

...

3. $BS \rightarrow H$: intruder ids

Additional notation:

d :	Sensor data
V :	An array of node ids
ka :	Symmetric key shared by A and BS
ka' :	MAC key shared by node A and BS
ca :	Counter shared by node A and BS
$E_{ka,ca}(d)$:	Data encryption using ka and ca
$D_{ka,ca}(d)$:	Decryption using ka and ca
$\mathsf{mac}_{ka'}$(msg) :	MAC calculated using ka'
$f()$:	One-way hash function
add(h, V) :	Add id h to V
store(h) :	Store id h for future validation

Figure 2: Secure-LEACH protocol

The BS keeps S secret, but shares the last element k_n with the rest of the network.

4.3.2. Setup Phase

Advertisement

Once it decides to be a CH, a node H broadcasts a sec_adv message (message 1, Fig. 2), which is a concatenation of its own id with a MAC value produced using the MAC key it shares with the BS. Ordinary nodes simply 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 (message 2) using μTESLA, and k_j. k_j is disclosed after a certain time period (message 3), after all nodes in the network have received the previous message.

Cluster Joining

After receiving both the broadcast and the corresponding key, the 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 $f(k_j) = k_{j+1}$.) The node then chooses a CH from this list using the original algorithm (based on signal strengths), and sends the join_req message (message 4) to the CH it chooses. Note that this message is unprotected, and identical to message 2, Fig. 1

Confirmation

After the CHs receives all the join_reqs, they broadcast the TDMA schedule to their cluster members (message 5).

4.3.3. Steady-state Phase

During this phase, sensor nodes send sensor measurements to their CHs, who then forward them (with or without data fusion) to the BS. We protect this traffic using SNEP, which encrypts and produces the MAC for the plain message. The BS will discard any messages that it cannot authenticate, and will see their originators as intruders. As soon as these intruder nodes are detected, the BS reports their identities to the CHs, who will then drop message from these nodes for the remaining of the round.

4.4. Security Analysis

CHs play a critical role in cluster-based communication protocols. Because they process and route sensor readings from a large number of sensor nodes to the BS, they can disrupt

whole regions of the network, if they misbehave. Our solution allows authentication of sec_adv messages, and prevents intruders from becoming CHs. Thus, unless sensor nodes are compromised (we do not address attacks by insiders in this work), the network is protected against selective forwarding, sinkhole, and HELLO flood attacks [5].

Our solution does not prevent intruders from joining the clusters. From the point of view of message origin authenticity, this prevention is not strictly necessary, given that the sensor data sent by intruders during the stead-state phase will not be authenticated by the BS, and will be discarded. The intruder will not succeed if its goal is energy depletion of its CH either (through forwarding bogus messages to the BS), because it can be readily flagged by the BS, and its CH will cease to forward its messages. Also, they can try to become members of a cluster to crowd its TDMA schedule, and disturb the communication within the cluster. But this can be accomplished by other much easier means, such as jamming the communications channels, for example.

Finally, all communication between the nodes and the BS is encrypted, authenticated, and protected against message tampering and replay.

5. Discussion

Our solution is extremely simple: each node, aside from the BS, is preloaded with only two keys, one for secure end-to-end communication with the BS, and the other for authenticating broadcasts from the BS.

5.1. Efficiency

Our solution for securing the protocol setup is quite efficient. The only secure-LEACH messages (Fig. 2) not found in the original LEACH protocol (Fig. 1) are the authenticated broadcast (message 2), whose length is determined by the number of CHs that there will be in the round, and the key disclosure (message 3) from the BS. Because the BS is resource-rich, these broadcasts and one single symmetric key encryption will not strain it.

In terms of the sensor nodes, CHs now send sec_adv instead of adv. This incurs each CH a MAC computation and some additional 64 bits for the MAC code in their sec_adv broadcast. As for the non-CH nodes, the additional work required of them has to do with receiving and processing the BS's authenticated broadcast (steps 2 and 3, Fig. 2). The length of message 2 is determined by the number of CHs that the BS managed to authenticate, which may vary slightly in each round. In any case, there is only one such message per round, so we expect the cost to be tolerable (though a deeper analysis would be required for a definite conclusion). To process message 2, each node needs to compute a MAC, and one (a few in cases where desynchronization occurs) application of f. This cost is also minimum.

For the steady-state phase, our solution encrypts messages from the sensor nodes to the BS using SNEP. This incurs one symmetric key encryption per message at the sensor nodes, and an additional 64-bits of MAC code for transmission, which is quite reasonable. Because the CHs do not share keys with its cluster members, they are unable to decrypt the messages from these members. The only thing they can do is to forward them. While this transfers the burden of decryption to the BS (which is something desirable), it prevents the CHs from doing data fusion, thus increasing their consumption

with communication. This increase will depend on the type of sensor data that is being collected and the type of data fusion (or aggregation) being carried out. In some cases it would be desirable to allow data fusion and we present a solution at subsection 5.5.

5.2. LEACH with additional control messages

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.

5.3. Applicability

As was observed before, our solution is quite efficient and does not require major modifications or additions to the original protocol. This was possible because no extra communication was needed to transmit the sec_adv messages to the BS, which was assumed to be within every node's communication range by the original LEACH protocol. We expect our solution to be applicable to any cluster-based communication protocols where this assumption holds.

5.4. Adding new nodes to the network

Sometimes it is necessary to add new nodes to the network, to replace nodes that died, or to extend the area being monitored. It is straightforward to do security provisioning using our scheme. For each new node A, we generate a symmetric key k_A, which will be preloaded to A, and communicated to the BS. k_A will be the secret key shared between A and the BS. A also needs to know the key k_i that was used to protect the last authenticated broadcast from the BS. There are a few simple ways how this can be accomplished: (1) the BS can send it encrypted to A, using the secret key they share; (2) A can be preloaded with any value k_j in S that has been disclosed, and the authenticity of the next key to be disclosed k_{i+1} can be checked by applying the one-way function f successively to k_{i+1} until k_j is obtained, since $j > i + 1$.

5.5. Data Fusion

Trading off security with efficiency, our basic solution does not allow data fusion by CHs, which would require shared keys between a CH and its cluster members. One straightforward solution would be to use the BS as a key distribution center. In networks with a large number of sensor nodes, this solution would consume a rather high amount of energy with key generation and distribution.

To allow data fusion, CHs need to decrypt messages. In the simplest solution, $n-1$ keys for each node would be needed since every node is a potential CH. We propose a cheaper solution that uses one distinct cluster key for each cluster. This key is generated at base station, delivered to CH during LEACH setup phase and are valid only for the next round. By avoid reusing cluster keys we make the system more robust against cryptanalysis attacks.

The BS will send cluster keys encrypted with each CH's secret key and this could be done right after message 2 in figure 2. Ordinary members of a cluster receive the key for the cluster they join, not at the setup phase but by demand. Before the first node

transmission in each round, it asks the BS for the cluster key in use for the current round. This allows every node and its CH to share the same key during each round.

This solution is more appropriate for those scenarios where sensor node transmissions are event-driven. In these scenarios, it is common for parts of the network to sense the occurrence of an event, and thus not every sensor node will transmit a report. On the other hand, the worst situation occurs when all nodes have some report to transmit right after the start of a new round. In these cases the BS would become a bottleneck, and this solution is likely to be impractical.

This solution make use of a relative small number of keys compared to the n^2 keys needed in the conventional key distribution scheme. When compared to the global key solution, cluster keys offer higher level of security because they are valid only for one round and for one cluster. The cost is more processing at the BS and more network traffic.

5.6. A Network's Lifetime x Length of the broadcast key chain

In our solution, the BS uses μTESLA to send authenticated broadcast. In each round, one key from a one-way key chain is used and disclosed. It would be ideal if the BS can be initialized with a chain that is long enough to cover the whole lifetime of the network. Given that BSs are usually high-power machines, storage space for long chains should not be an issue. If, however, the chain runs out before the network's time is over, a new chain T can be generated by the BS, and its last key k_n^t be authenticated using the first key k_0 of the original chain. The length of the original chain does not, thus, determine the lifetime of the network.

6. Conclusion

To the best of our knowledge, this is the first study that focuses on adding security to cluster-based communication protocols in homogeneous WSNs with resource-constrained sensor nodes. We proposed secure-LEACH, the first secure version of LEACH, using building blocks from SPINS. Secure-LEACH prevents an intruder from becoming a CH or injecting bogus sensor data into the network by pretending to be one of its members. It also encrypts and authenticates sensor data, and protects them from tampering and replay. Secure-LEACH is lightweight and preserves the structure of the original LEACH.

The simplicity of our solution relies on LEACH's assumption that every node can reach a BS by transmitting with sufficiently high power. This assumption will not always be true, and we are investigating alternatives in cases where it is not true.

References

[1] 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.

[2] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Research in Security and Privacy*, 2003.

[3] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *In Proc. of ACM CCS*, 2002.

[4] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS*, January 2000.

[5] Chris Karlof and David Wagner. Secure routing in wireless sensor network: Attacks and countermeasures. *First IEEE International Workshop on Sensor Network Protocols and Applications*, May 2003.

[6] Stephanie Lindsey and Cauligi S. Raghavendra. Pegasis: Power-efficient gathering in sensor information systems. In *IEEE Aerospace Conference*, March 2002.

[7] A. Manjeshwar and D. Agrawal. Teen: A routing protocol for enhanced effeciency in wireless sensor networks. In *1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, 2001.

[8] Leonardo B. Oliveira, Hao Chi Wong, Antonio A. Loureiro, and Daniel M. Barbosa. A security protocol for hierarchical sensor networks. In *Proceedings of the 2004 SBRC - Simposio Brasileiro de Redes de Computadores*, May 2004.

[9] Adrian Perrig, Robert Szewczyk, Victor Wen, David E. Culler, and J. D. Tygar. Spins: Security protocols for sensor networks. In *Mobile Computing and Networking*, pages 189–199, 2001.

[10] L. Venkatraman and D. Agrawal. A novel authentication scheme for ad hoc networks. In *IEEE Wireless Communications and Networking Conference (WCNC 2000), vol. 3,*, pages 1268–1273, 2000.

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