Available online at www.sciencedirect.com
ScienceDirect
Signal Processing 87 (2007) 2882–2895
SIGNAL PROCESSING
www.elsevier.com/locate/sigpro

SecLEACH—On the security of clustered sensor networks

Leonardo B. Oliveira[a,*], Adrian Ferreira[c], Marco A. Vilaça[c], Hao Chi Wong[b], Marshall Bern[b], Ricardo Dahab[a], Antonio A.F. Loureiro[c]

[a]*University of Campinas, Brazil*
[b]*Palo Alto Research Center, CA, USA*
[c]*Federal University of Minas Gerais, Brazil*

Received 25 August 2006; received in revised form 5 February 2007; accepted 15 May 2007
Available online 25 May 2007

Abstract

Clustered sensor networks have recently been shown to increase system throughput, decrease system delay, and save energy while performing data aggregation. Whereas those with rotating cluster heads, such as LEACH (low-energy adaptive clustering hierarchy), have also advantages in terms of security, the dynamic nature of their communication makes most existing security solutions inadequate for them. In this paper, we investigate the problem of adding security to hierarchical (cluster-based) sensor networks where clusters are formed dynamically and periodically, such as LEACH. For this purpose, we show how random key predistribution, widely studied in the context of flat networks, and μTESLA, a building block from SPINS, can be both used to secure communications in this type of network. We present our solution, and provide a detailed analysis of how different values for the various parameters in such a system impact a hierarchical network in terms of security and energy efficiency. To the best of our knowledge, ours is the first that investigates security in hierarchical WSNs with dynamic cluster formation.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Security; Hierarchical (cluster-based) sensor networks; Secure data aggregation

1. Introduction

Wireless sensor networks (WSNs) [1,2] are rapidly emerging as a technology for monitoring different environments of interest and they find applications ranging from battlefield reconnaissance to environmental protection. When embedded in critical applications, WSNs are likely to be attacked [3,4]. Aside from the well-known vulnerabilities due to wireless communication, WSNs lack physical protection and are usually deployed in open, unattended environments, which makes them more vulnerable to attacks. It is therefore crucial to devise security solutions to these networks.

An important issue one needs to tackle when using cryptographic methods to secure a network is key distribution, which has been intensively studied recently (e.g., [5–22]) in the context of WSNs. It is worth noting, however, that a large number [23] of WSN architectures have been proposed and a key distribution solution that is well suited to one architecture is likely not to be the best for another,

*Corresponding author.
E-mail addresses: leob@ic.unicamp.br (L.B. Oliveira), adrian@dcc.ufmg.br (A. Ferreira), vilaca@dcc.ufmg.br (M.A. Vilaça), hcwong@parc.com (H.C. Wong), bern@parc.com (M. Bern), rdahab@ic.unicamp.br (R. Dahab), loureiro@dcc.ufmg.br (A.A.F. Loureiro).

0165-1684/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.sigpro.2007.05.016

as different network architectures exhibit different communication patterns.

Cluster-based organization (e.g., [24,25]) has been proposed for ad hoc networks in general and WSNs in particular. In cluster-based networks, nodes are typically organized into clusters, with cluster heads (CHs) relaying messages from ordinary nodes in the cluster to the base stations (BSs). Clustered WSNs were first proposed for various reasons including scalability and energy efficiency while performing data aggregation. Those with rotating CHs, like LEACH (low-energy adaptive clustering hierarchy) [24], are also interesting in terms of security, as their routers (the CHs), which are more prominent targets for adversaries because of their role in routing, rotate from one node to another periodically, making it harder for an adversary to identify the routing elements and compromise them [3].

Adding security to LEACH-like protocols is challenging, as its dynamic (at random) and periodic rearranging of the network's clustering (and changing links) makes key distribution solutions that provide long-lasting node-to-node trust relationships (to be sure, provided by most existing solutions) inadequate.

In this paper, we focus on providing efficient security communications in LEACH-like protocols. To this end, we first propose SecLEACH, a modified version of LEACH that applies random key predistribution and μTESLA to provide baseline security. We then give a detailed analysis and performance evaluation of our scheme, and present concrete numbers on how the various parameters impact the tradeoffs between cost and security. To our knowledge, SecLEACH is the first solution to secure hierarchical (cluster-based) WSNs with dynamic cluster formation. Our main contributions in this paper are:

1. to have provided an efficient solution for securing communications in LEACH and
2. to have shown how random key predistribution and μTESLA can be used to secure hierarchical WSNs with dynamic cluster formation.

To be sure, random key predistribution has been studied profusely [15], but always in the context of flat WSNs. Due to this fact, these studies have not taken into consideration communication patterns of hierarchical (cluster-based) networks and thus cannot be applied, as is, to them. To the best of our knowledge, ours is the first that investigates random key predistribution as applied to hierarchical (cluster-based) WSNs with rotating CHs.

The rest of this paper is organized as follows. In Section 2, we discuss what is needed to cryptographically secure LEACH's communications and why existing solutions are inadequate. We present our solution (SecLEACH) in Section 3, and analyze its performance in Section 4. Finally, we discuss related work and conclude in Sections 5 and 6, respectively.

2. Adding security to LEACH

WSNs typically comprise of one or more BSs and a larger number of resource-scarce sensor nodes. Sensor nodes do not typically communicate directly with the BS because: (1) they typically have transmitters with limited transmission range, and are unable to reach the BS directly and (2) even if the BS is within a node's communication range, direct communication typically demands a much higher energy consumption.

Multi-hop flat networks are a more energy-efficient alternative, that has a node take advantage of its neighboring nodes as routers: farther away nodes send their messages to intermediate nodes, which then forward them toward the BS in a multi-hop fashion. The problem with this approach is that, even though peripheral nodes actually save energy, the intermediate nodes spend additional energy receiving and forwarding messages, and end up having a shortened lifetime.

LEACH (low-energy adaptive clustering hierarchy) [24] was proposed to address the aforementioned problem. It assumes that every node can directly reach a BS by transmitting with high enough power. However, to save energy, sensor nodes send their messages to their CHs, which then aggregate the messages, and send the aggregate to the BS. To prevent energy drainage of a restricted set of CHs, LEACH randomly rotates CHs among all nodes in the network, from time to time, thus distributing aggregation- and routing-related energy consumption among all nodes in the network.

LEACH thus works in rounds. In each round, it uses a distributed algorithm to elect CHs and dynamically cluster the remaining nodes around the CHs. The resulting clustering structure is used by all sensor-BS communications for the remaining of the round.

Using a set of 100 randomly distributed nodes (and a BS located at 75 m from the closest node), simulation results [24] show that LEACH spends up to 8 times less energy than other protocols. To be fair, the energy saving comes from a number of sources other than just dynamic cluster-based communication: data aggregation (CHs aggregate data 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 for a reasonable amount of time), and transmitter calibration (nodes calibrate their transmitters' power in such a way that they are only high enough to reach the CH).

2.1. LEACH: 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 Step 1 (*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 decide to do so broadcast a message (adv) advertising this fact, at a level that can be heard by everyone in the network. To avoid collision, a carrier sense multiple access protocol is used. In Step 2 (*cluster joining* step), the remaining nodes pick a cluster to join based on the largest received signal strength of an adv message, and communicate their intention to join by sending a join_req (join request) message. Once the CHs receive all the join requests, Step 3 (*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. Given that all transmitters and receivers are calibrated, balanced and geographically distributed clusters should result.

Once 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 (Step 4), 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 (Step 5). The steady-state phase consists of multiple reporting cycles, and lasts much longer compared to the setup phase.

2.2. Security vulnerabilities

Like most routing protocols for WSNs, LEACH is vulnerable to a number of security attacks [3], including jamming, spoofing, replay, etc. However, because it is a cluster-based protocol, relying fundamentally on the CHs for data aggregation and 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 workings of 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.

It is worth noting that LEACH is more robust against insider attacks than most other routing protocols [3]. In contrast to more conventional multihop schemes where nodes around the BS are

Setup phase

1. $H \Rightarrow \mathcal{G}: \quad id_H, \mathsf{adv}$
2. $A_i \rightarrow H: \quad id_{A_i}, id_H, \mathsf{join\_req}$
3. $H \Rightarrow \mathcal{G}: \quad id_H, (\ldots, \langle id_{A_i}, t_{A_i} \rangle, \ldots), \mathsf{sched}$

Steady-state phase

4. $A_i \rightarrow H: \quad id_{A_i}, id_H, d_{A_i}$
5. $H \rightarrow BS: \quad id_H, id_{BS}, \mathcal{F}(\ldots, d_{A_i}, \ldots)$

The various symbols denote:

A_i, H, BS :	An ordinary node, a cluster head, and the base station, respectively	d_X :	Sensing report from node X
$\mathcal{G}$:	The set of all nodes in the network	$\langle id_X, t_X \rangle$:	Node X's id and its time slot t_X in its cluster's transmission schedule
$\Rightarrow, \rightarrow$:	Broadcast and unicast, transmissions respectively	adv, join_req, sched :	String identifiers for message types
id_X :	Node X's id	$\mathcal{F}$:	Data aggregation function

Fig. 1. LEACH protocol.

especially attractive for compromise (because they concentrate all network-to-BS communication flows), CHs in LEACH communicate directly with the BS, can be anywhere in the network, and change from round to round. All these characteristics make it harder for an adversary to identify and compromise strategically more important nodes.

2.3. *Why existing key distribution schemes are inadequate*

One of the first steps to be taken to secure a WSN is to prevent illegitimate nodes from participating in the network. This access control can preserve much of a network's operations, unless legitimate nodes have been compromised. (Note that access control does not solve all security problems in WSNs. E.g., it is ineffective against DoS attacks based on jamming wireless channels, or manipulating a node's surrounding environment to induce the reporting of fabricated conditions.) Access control in networks has typically been implemented using cryptographic mechanisms, which rely critically on key distribution. Key distribution is thus of paramount importance in securing a network.

There are a number of standard key distribution schemes in the security literature [26], most of which are ill-suited to WSNs: public key based distribution, because of its processing requirements; global keying, because of its security vulnerabilities; complete pairwise keying, because of its memory requirements; and those based on a key distribution center, because of its inefficiency and energy consumption [8].

Some key distribution schemes (e.g., [6,8,12,13,15,19,20]) have been specifically designed for WSNs. While they are well-suited for network organizations they were designed for, they are inadequate for other organizations. These schemes typically assume that a node interacts with a quite static set of neighbors and that most of its neighborhood is discovered right after the deployment. However, clusters in LEACH are formed dynamically (at random) and periodically, which changes interactions among the nodes and requires that any node needs to be ready to join any CH at any time.

For instance, Zhu et al. [8] (LEAP) and Eschenauer and Gligor's [6] schemes are rather efficient for flat networks where nodes interact with a rather static set of neighbors, but are inadequate as is for LEACH's periodic rearranging of the network. For example, if LEAP were used to secure communication in LEACH, a new key distribution could be required per round. This not only would be inefficient, but also infeasible, as LEAP relies on a master key to perform key distribution, which is erased from the nodes' memory as soon as the first key distribution is completed. Similarly, Eschenauer and Gligor's scheme, based on random keys, does not provide mechanisms to authenticate broadcasts from CHs to the rest of the network (Fig. 1, Steps 1 and 3). Such authentication is essential to secure the periodic (re)clustering procedure. Eschenauer and Gligor's scheme will be explained in more detail in Section 3.2.

In what follows, we discuss the network model assumed in LEACH, and the requirements it sets for key distribution.

2.4. *Key distribution for LEACH: requirements and constraints*

Our discussion in Section 2.2 shows the need for the nodes to authenticate each other as legitimate members of the network both in the setup interactions and the sensor data reporting communications. Given the communication patterns in LEACH, two different types of authentication are required: authenticated broadcast, for broadcasts from the CHs to the rest of the network (Fig. 1, Steps 1 and 3); and pairwise authentication for the remaining (node-to-CH and CH-to-BS) communications.

Symmetric-key authenticated broadcasts for WSNs, both global (μTESLA [27]) and local (LEAP [8]), share the core idea of using a one-way key chain (a sequence of keys $k_1, \dots k_n$, where k_{i+1} is generated from k_i by applying a one-way hash function $f()$, i.e., $k_{i+1} = f(k_i)$) to achieve authentication. These schemes cannot be applied, as is, to LEACH because: (1) the key chain would require significant storage space in the broadcasting CHs and more importantly, (2) all nodes in the network would need to store one key for each node in the network, which is neither practical nor scalable. (Each node needs to store one key for every other node in the network because an ordinary node needs to be able to authenticate the CHs in each round, which can be arbitrary nodes in the network.)

Pairwise authentication is also challenging to implement in LEACH, because of key distribution issues. Effectively, given that any node needs to be ready to join any CH (which could be any node in the network), it would need to have shared pairwise keys with every other node in the network. Just like in authenticated broadcast, this is neither practical, nor scalable.

3. SecLEACH—Applying random key distribution to LEACH

In this section, we first briefly describe the main ideas behind μTESLA (Section 3.1) and random key predistribution schemes (Section 3.2), then we show how they can be used to secure LEACH (Section 3.3). We note that we use LEACH to be concrete, but that our proposal should have a wider applicability, and be easily adaptable to other similar protocols.

3.1. μTESLA

μTESLA was proposed by Perrig et al. in SPINS [27] and provides authenticated broadcast. The protocol implements the asymmetry required for authenticated broadcast using one-way key chains constructed with cryptographically secure hash functions, and delayed key disclosure. In μTESLA each node X is assigned a group key k^n that is shared by all members of the network. k^n is the last key of a sequence $\mathscr{J}$ generated by applying successively a one-way hash function f to an initial key k^0 ($\mathscr{J} = k^0, k^1, k^2, \ldots, k^{n-1}, k^n$, where $f(k^j) = k^{j+1}$). The BS keeps $\mathscr{J}$ secret, but shares the last element k^n with the rest of the network. After the deployment, whenever the BS wants to broadcast a message it identifies the last key k^j in J that has not been disclosed, produces a MAC[1] of the message using k^j, and sends both the message and the MAC. k^j is disclosed after a certain time period, after all nodes in the network have received the previous message. After receiving both the broadcast and the corresponding key, nodes in the network can authenticate the broadcast from the BS by checking if the key is a an element of the key chain generated by the BS, and immediately precedes the one that was released last. That is, if $f(k^j) = k^{j+1}$. μTESLA requires loose time synchronization. See SPINS [27] for further details on μTESLA.

[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.

3.2. Random key predistribution schemes

Random key predistribution for WSNs was first proposed by Eschenauer and Gligor [6], and has since been studied by several research groups [15]. In a random key predistribution scheme, each node is assigned a set of keys drawn from a much larger key pool. Different schemes have different assignment algorithms, but they all result in probabilistic key sharing among the nodes in the network.

To bootstrap security using Eschenauer and Gligor's original scheme [6], a network goes through three phases. In the first phase (*key predistribution*), which takes place prior to network deployment, a large pool of S keys and their ids are generated. Each node is then assigned a ring of m keys, drawn from the pool at random, without replacement. In the second phase (*shared-key discovery*), which takes place during network setup, all nodes broadcast the ids of the keys on their key rings. Through these broadcasts, a node finds out with which of their neighbors (as determined by communication range) they share a key. These keys can then be used for establishing secure links between the two neighbors. Finally, during *path-key establishment* phase, pairs of neighboring nodes that do not share a key can set up their own keys, as long as they are connected by two or more secure links at the end of shared key discovery.

Because of the way keys are assigned, a key can be found in more than two nodes, and used in multiple communication links. When a node is compromised, all its keys are compromised, and all the links secured by these keys are also compromised.

The initial assignment of key rings to nodes can also be done pseudorandomly [12,13]. Pseudorandom schemes make both the key predistribution and the shared-key discovery more efficient.

3.3. SecLEACH: protocol description

In our solution, we propose to generate a large pool of S keys and their ids prior to network deployment. Each node is then assigned a ring of m keys drawn from the pool pseudorandomly [12], without replacement, as follows. For each node X, we use a pseudorandom function (PRF) to generate its unique id id_X. id_X is then used to seed a pseudorandom number generator (PRNG) of a large enough period to produce a sequence of m numbers. $\mathscr{R}_X$, the set of key ids assigned to X, can then be obtained by mapping each number in the sequence to its correspondent value *modulus s*. Also prior to deployment, each node is assigned a pairwise key shared with the BS; and a group key (the last key of a one-way key chain held by the BS) that is shared by all members of the network.

The LEACH clustering algorithm can then be run with the following modifications. When a self-elected CH broadcasts its adv message, it includes information of the keys in its key ring. The broadcast is authenticated leveraging on the BS, who is trusted and has more resources. The remaining nodes now cluster around the closest CH with whom they share a key. Fig. 2 shows the details of our SecLEACH protocol.

In Step 1, a self-elected CH H broadcasts (Steps 1.1) its id id_H, a nonce, and a MAC produced using the key the CH shares with the BS (which will be used by the BS for the purpose of authentication). The BS waits to hear and authenticate (modified) adv messages from all CHs; compiles the list of legitimate CHs; and sends the list to the network using μTESLA (Steps 1.2 and 1.3). Ordinary nodes now know which of the adv messages 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.

In Step 2, ordinary nodes A_i compute the set of H's key ids (using the pseudorandom scheme described above), choose the closest CH with whom they share a key $k_{[r]}$, and send it a join_req message, protected by a MAC. The MAC is generated using $k_{[r]}$, and includes the nonce from H's broadcast in Step 1 (to prevent replay attacks), as well as the id r of the key chosen to protect this link (so that the receiving CH knows which key to use to verify the MAC).

In Step 3, to conclude the setup phase, the CHs broadcast the time slot schedule to the nodes that chose to join their clusters. This broadcast is authenticated the same way as the previous one (Step 1.1). For clarity of presentation, we do not reproduce the full-blown authenticated version here.

In the steady-state phase, node-to-CH communications (Step 4) are protected using the same key used to protect the join_req message in Step 2. A value computed from the nonce (*nonce*) and the reporting cycle (l) is also included to prevent replay. The CHs can now check the authenticity of sensing reports they receive, perform data aggregation, and send the aggregate result to the BS (Step 5). The aggregate result is protected using the symmetric key shared between the CH and the BS. For freshness, a counter (shared between the CH and

Setup phase

1.1.	$H \Rightarrow \mathcal{G}$:	$id_{A_i}, id_H, \mathsf{mac}_{k_H}(id_H \mid c_H \mid \mathsf{adv})$
	A_i :	$\mathsf{store}(id_H)$
	BS :	if $\mathsf{mac}_{k_H}(id_H \mid c_H \mid \mathsf{adv})$ is valid, $\mathsf{add}(id_H, \mathcal{V})$
1.2.	$BS \Rightarrow \mathcal{G}$:	$\mathcal{V}, \mathsf{mac}_{k^j}(\mathcal{V})$
1.3.	$BS \Rightarrow \mathcal{G}$:	k^j
	A_i :	if $\left(f\left(k^j\right) = k^{j+1}\right)$ and $(id_H \in \mathcal{V})$, H is authentic
	A_i :	choose r such that $r \in (\mathcal{R}_H \cap \mathcal{R}_{A_i})$
2.	$A_i \rightarrow H$:	$id_{A_i}, id_H, r, \mathsf{join\_req}, \mathsf{mac}_{k_{[r]}}(id_{A_i} \mid id_H \mid r \mid nonce)$
3.	$H \Rightarrow \mathcal{G}$:	$id_H, (\ldots, \langle id_{A_i}, t_{A_i}\rangle, \ldots)$, sched

Steady-state phase

4.	$A_i \rightarrow H$:	$id_{A_i}, id_H, d_{A_i}, \mathsf{mac}_{k_{[r]}}(id_{A_i} \mid id_H \mid d_{A_i} \mid nonce + l)$
5.	$H \rightarrow BS$:	$id_H, id_{BS}, \mathcal{F}(\ldots, d_{A_i}, \ldots), \mathsf{mac}_{k_H}(\mathcal{F}(\ldots, d_{A_i}, \ldots) \mid c_H)$

Symbols as previously defined, with the following additions:

r :	Id of a key in the key ring	$f()$:	One-way hash function
$\mathcal{R}_X$:	Set of key ids in node X's key ring	$\mathsf{mac}_k(\mathsf{msg})$:	MAC calculated using key k
$\mathcal{V}$:	A set of CH ids	c_X :	Counter shared by node X and BS
k_X :	Symmetric key shared by node X and BS	l :	Reporting cycle within the current round
k^j :	j-th key in a one-way key chain;	$\mathsf{store}(id_H)$:	Store id_H for future validation
$k_{[r]}$:	Symmetric key associated with id r	$\mathsf{add}(id_H, \mathcal{V})$:	Add id_H to $\mathcal{V}$

Fig. 2. SecLEACH protocol.

the BS) is included in the MAC value as well. Fig. 2 shows only one reporting cycle in the steady-state phase. In practice, there will be multiple cycles in a round. In each round l, the value of the "freshness token" (denoted by "$nonce + l$" in Step 4, and "c_H" in Step 5) needs to be incremented by 1.

At the end of the clustering process, we expect that a fraction of the ordinary nodes will be matched with a CH, though not necessarily the one they would have matched with in the basic LEACH, because of key sharing constraints; the remaining would not have any CH to match with. We call these nodes orphans.

There are different ways to deal with the orphans: we can have them sleep for the round; we can add a small protocol that would allow the "already-adopted children" to bring the orphans into their clusters; or we can have them communicate directly with the BS for the round. In any case, the number of orphans will depend on the size of the key pool, the size of the key ring, and the number of CHs, and will have an impact on the performance of the network.

In Section 4, we show the cost, efficiency, and security of SecLEACH, as well as the tradeoffs when we vary the various parameter values.

3.4. Security analysis

SecLEACH provides authenticity, integrity, confidentiality, and freshness to communications. Our solution allows authentication of adv messages (steps 1.1, 1.2, and 1.3, Fig. 2), and prevents unauthorized nodes from becoming CHs and in turn adopting nodes. The message in Step 2, Fig. 2, is protected with a key in the key pool; and a successful check of this message allows H to conclude that the message originated from a legitimate node in the network. Because the protected message includes the nonce from Step 1, H can also conclude that it is not a stale message being replayed. The same observations apply to the message in Step 4. The freshness of all subsequent sensor reports from the ordinary nodes to their CHs is guaranteed by nonces values that are incremented each time. For the message in Step 5, the freshness is guaranteed by the counter value shared between the CH and the BS; the counter value also being incremented each time the CH sends a new report to the BS. Remember (3.3) that sched messages (Step 3) are authenticated the same way as adv messages, but for clarity of presentation we do not reproduce the full-blown authenticated version in Fig. 2.

Because link keys used for node-to-CH communications are not pairwise in SecLEACH (i.e., a number of other nodes other than the end points of a compromised link may have the key used in the link), the biggest security issue in SecLEACH is likely to be its resiliency against node captures. We discuss this issue in Section 4.2.

4. Evaluation of our scheme

Random key predistribution schemes have all been introduced and studied in the context of flat networks, which come with the following assumptions: (1) the nodes have antennas with limited transmission range and (2) node-to-BS communications are multi-hop, with the nodes each relying on their neighbors to forward messages towards the BS. In this context, any two nodes within each other's transmission range has a communication link between them, and a forwarding route can be established between any two nodes (including the BS) as long as one can overlay a connected graph on the network using these range-defined links.

In flat networks where security is to be bootstrapped from random key predistribution, there is a (secure) link between two nodes only if they are within each other's communication range and share a key. In this new context, a (secure) forwarding route can be established between any two nodes (including the BS) only if one can overlay a connected graph on the network using secure links. Given that it is possible that a physical (range-defined) link will be (logically) severed by lack of a shared key between the two end nodes, one needs to choose the parameters S (size of the key pool) and m (size of the key ring) in such a way that the resulting network is still (securely) connected, with high probability.

In the context of LEACH, the assumptions are slightly different: (1) any node in the network is reachable from any other node in single hop; but (2) node-to-BS communications are typically carried out in two-hops: from ordinary nodes to CHs, and from CHs to the BS. Because of the first assumption, any ordinary nodes can theoretically join any CH; in practice, they choose the closest to save energy. For energy efficiency, however, a network needs to use just the right number of CHs, as different number of CHs leads to different energy consumptions.

In SecLEACH, because of the constraints imposed by key sharing, not all CHs are accessible to all ordinary nodes. In fact, depending on the values of S and m, which determine the probability that two nodes will share a key, an ordinary node will have a larger or smaller number of CHs to choose from. To achieve maximum energy efficiency in the context of SecLEACH, therefore, one needs to find right values for S, m, and the number of CHs. In what follows, we show how different parameter values impact a network, in terms of security and energy efficiency.

4.1. Parameter values and their impact on performance

Given a WSN, the amount of storage reserved for keys in each node is likely to be a preset constraint, which makes the size of the key ring m a fixed parameter in the system. Once m is set, the choice of S will impact the system in two ways:

1. *Its security level*: Given a (S, m)-network, a network where each node is assigned m keys from a key pool of size S, m/S is the probability that a randomly chosen link will be compromised when a node that is not either end of the link is compromised. The *security level* sl of a (S, m)-network can then be defined as

   $$\text{sl} = 1 - \frac{m}{S},$$

   which gives the probability that a randomly chosen link is not compromised when a node that is not either end of the link is compromised. Note that given a fixed m, the larger the S, the larger the sl (the higher the security level).
2. *The probability that two nodes will share a key*: Given any two nodes in an (S, m)-network, the probability P_s that they will share a key is given by[2]

   $$P_s = 1 - P_{\bar{s}},$$

   where $P_{\bar{s}}$, the probability that they will not share a key, is given by

   $$P_{\bar{s}} = \frac{[(S-m)!]^2}{S!(S-2m)!}.$$

   Note that given a fixed m, the larger the S, the smaller the P_s.

The number h of CHs in the network is another parameter in the system. In LEACH, the density of CHs in a network determines the average distance between a node and its closest CH. This distance, in turn, determines the amount of energy needed in node-to-CH communications: the denser the CHs, the shorter the average node-to-CH distance, and the smaller the energy consumption for node-to-CH communications. On the other hand, CHs communicate with the BS in single hop. Thus, the larger the number of CHs, the more nodes will be communicating single-hop with the BS, and the more energy will be spent. Taking this reasoning into account, one can find an optimal value for h, which minimizes the total energy consumption, and maximize the network's lifetime.

In SecLEACH, only a fraction of h CHs is probabilistically accessible (as determined by key sharing) by an ordinary node. That is, h is actually a *nominal* value; what ultimately matters is the *effective* value, h_e, given by $h_e = h \times P_s$. Note that, to obtain a given h_e, one does not need to start with a fixed h. In fact, one can first fix a value for P_s, and adjust h accordingly.

P_s and h will also determine the expected orphan rate, that is, the probability that an ordinary node will be orphan. Given P_s (and consequently $P_{\bar{s}}$) and h, the expected orphan rate P_o is given by $P_o = (P_{\bar{s}})^h$. In a network with n nodes, it is then expected that $n \times P_o$ nodes will be orphans, and communicating single-hop with the BS. Fig.3 shows P_o as function of h under an sl $= 0.99$. Because P_o depends of the *absolute* number of the CHs, no

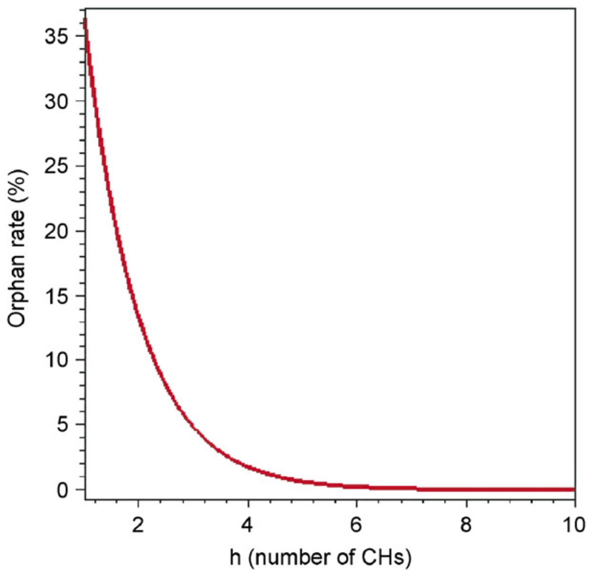

Fig. 3. Orphan rate, for sl = 0.99.

[2]This derivation was first shown in [6].

matter the network size n, the ratio of orphan nodes will be negligible for $h \geqslant 7$.

To show some concrete numbers and the tradeoffs induced by different parameter values, we provide estimates on energy consumption levels for different scenarios. For our estimates, we assume a network as in LEACH original paper, i.e., $n = 100$ nodes, uniformly distributed at random in a $10^4\,\mathrm{m}^2$ square area; and a BS located at the center of the square. We consider three key ring sizes for a fixed sl value ($m = 50, 100, 150$, for sl $= 0.99$) and three security levels for a fixed m value (sl $= 0.95, 0.98, 0.99$, for $m = 100$). Table 1 exhibits the respective P_s values for these scenarios. In each case, we take into account only the energy consumed for communication in the steady-state phase since we expected that setup overhead will be amortized among the multiple cycles of the subsequent steady-state phase. In addition, we do not consider the cost incurred by the cryptographic operations, as the operations we use have been shown [27] to incur a very small overhead compared to that incurred by communication.

To estimate the energy consumption, we assume the same radio energy model used in LEACH [24]. In this model, a radio dissipates $\varepsilon_\mathrm{r} = 50\,\mathrm{nJ/bit}$ to run the transmitter or receiver circuitry, and $\varepsilon_\mathrm{a} = 100\,\mathrm{pJ/bit/m}^2$ for the transmitter amplifier. Also, the radios expend the minimum required energy to reach the recipients and are turned off to avoid receiving unintended transmissions. An δ^2 energy loss due to channel transmission is assumed as well. Under this model, the costs to transmit ($\mathscr{E}_\mathrm{T}$) and receive ($\mathscr{E}_\mathrm{R}$) a β-bit message at distance δ, and the amount of energy $\mathscr{E}_\mathrm{cycle}$ the network consumes to go through one cycle of sensor data reporting are given, respectively, by:

$$\mathscr{E}_\mathrm{T}(\beta, \delta) = \beta\varepsilon_\mathrm{r} + \beta\delta^2\varepsilon_\mathrm{a},$$
$$\mathscr{E}_\mathrm{R}(\beta) = \beta\varepsilon_\mathrm{r},$$
$$\mathscr{E}_\mathrm{cycle} = (n-h)[\mathscr{E}_\mathrm{T}(\beta, \delta_1) + \mathscr{E}_\mathrm{R}(\beta)] + h\mathscr{E}_\mathrm{T}(\beta, \delta_2),$$

Table 1
Prob. P_s of key sharing as a function of security level sl and key ring size m

sl	m		
	50	100	150
0.95	–	0.995	–
0.98	–	0.870	–
0.99	0.396	0.636	0.780

where δ_1 is average distance between an ordinary node and its closest CH, and δ_2 is the average distance between a CH and the BS.

In what follows, we calculated δ_1 (Fig. 4) and δ_2 by using the derivation in Appendix A. Also, we set SecLEACH messages to be 36 bytes long (the default TinyOS message size [28]) and LEACH messages to be 30 bytes long. The difference is meant to account for the size difference between the MAC (8 bytes [27]) and CRC (2 bytes [29])—the former present in SecLEACH, but absent in LEACH; and the latter present in LEACH, but absent in SecLEACH.

Using the energy consumption model above, we obtained the energy consumption level for the various scenarios we considered. In Figs. 5 and 6, the values are for one cycle of sensor data reporting in the steady-state phase. Fig. 5 shows the energy consumption in node-CH communication for different security levels. Note that the consumption level is smaller in LEACH than in any instantiations of SecLEACH, and larger values of sl lead to larger overheads. On the other hand, the higher the h, the smaller the overhead. For a given security level, larger key rings decrease the energy consumption (Fig. 6). Note that, in all cases, there is a value of h for which the energy consumption is minimum.

We also estimated how scalable SecLEACH is. Table 2 shows the overhead incurred by SecLEACH, under the various parameter values and under different network sizes n, as compared to

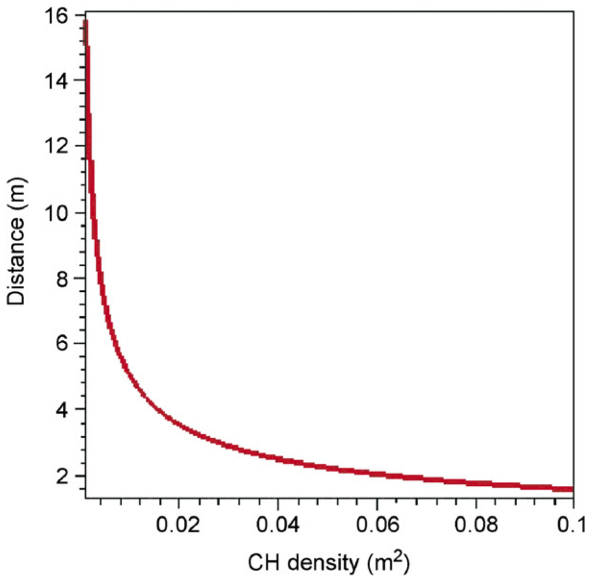

Fig. 4. Average node-to-CH distance.

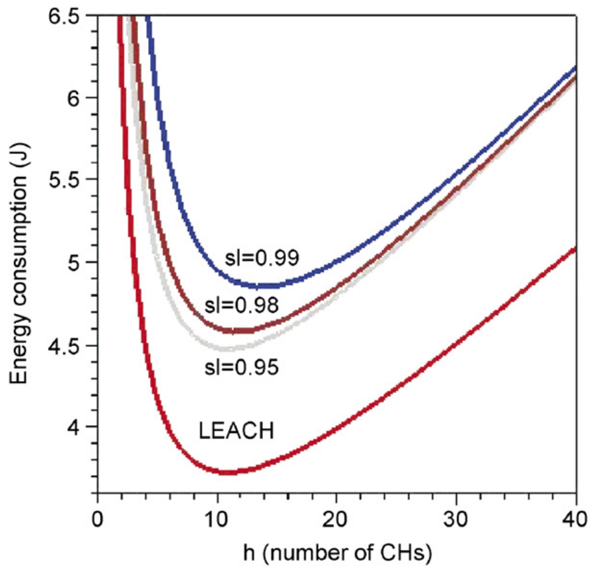

Fig. 5. Energy consumption, for $m = 100$.

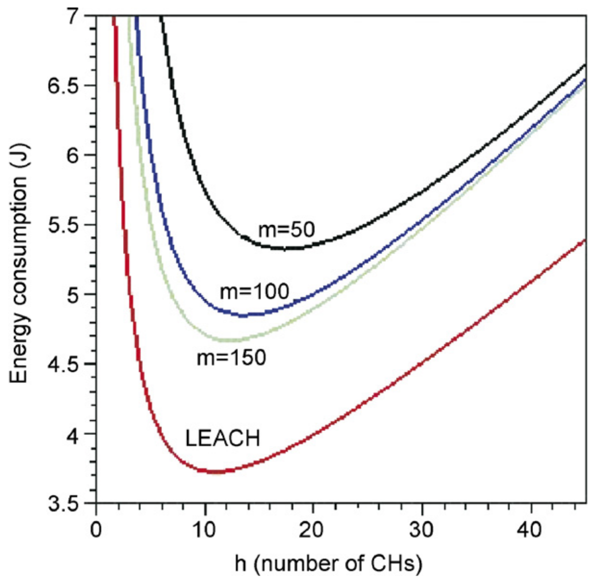

Fig. 6. Energy consumption, for sl = 0.99.

LEACH. In the estimates, we assume a constant node density (i.e., the larger the n, the larger the network area, as well) and a single BS. The overheads were computed using the values of h for which the energy consumption, in each scenario, is minimum. It is worth mentioning that overhead in SecLEACH is due to two factors: the increased message size (20% larger) and the increased node-CH distance—the CH–BS distance in SecLEACH is not increased as compared to LEACH, as every CH shares a key with the BS.

Table 2
Energy overhead

n	sl ($m = 100$)			m (sl = 0.99)		
	0.95	0.98	0.99	50	100	150
100	20.1%	22.9%	30.0%	42.9%	30.1%	25.3%
1000	20.2%	25.6%	39.8%	65.6%	39.8%	30.3%
10000	20.3%	27.4%	46.1%	80.6%	46.1%	33.6%

Note that for the maximum security level (sl = 0.99), the overhead increases from 30.0% to 46.1%, as the network becomes larger. This increase is due to an increase in the average CH–BS distance, caused by the increase in the size of the network. (Larger distances lead to more expensive communications.) To counterbalance this factor, the network could instantiate a smaller number of CHs. But this would increase the number of nodes performing node-CH communication (the one that incurs overhead) leading to an overall overhead increases as well.

This overhead, however, can be mitigated by using a larger values of m, as shown in the column $m = 150$. Alternatively, one may also choose to live with a lower security level. E.g., the P_s value for sl = 0.95 is very close to that for LEACH (Table 1), and the overhead in this case being due mainly to the increase in the message size. For such sl value, the solution is very scalable.

4.2. Resiliency against node capture

In key distribution schemes, resiliency against node capture measures how much of the network (its communication links) is compromised when a node is compromised. It is a critical performance measure that gauges the robustness of a solution. In SecLEACH, the values of m and S determines the probability that a random link will be compromised when a node (that is not either end of the link) is compromised.

The resiliency of random key predistribution has been studied before [7] in the context of flat networks and the same analysis is applicable in our context. Fig. 7 shows the percentage P_c of compromised links as a function of the absolute number of compromised nodes for the considered security levels. Note that P_c increases as the absolute number of compromised nodes (instead

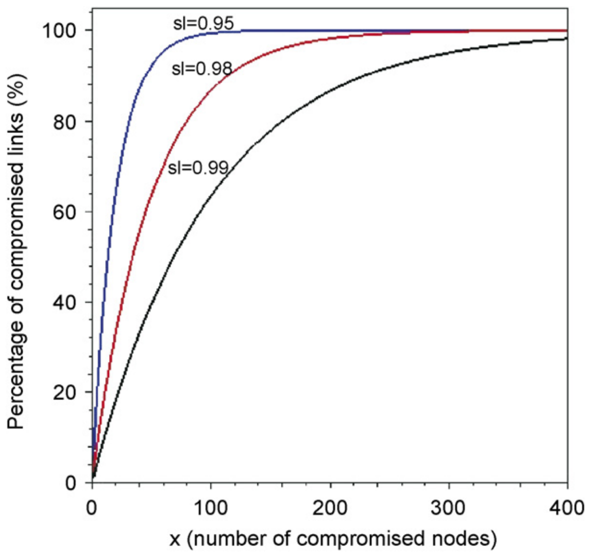

Fig. 7. SecLEACH's resiliency against node capture.

of percentage of nodes in the network) increases. Beyond a certain value of x, P_c reaches the value 1.0, no matter what is the security level. For smaller values of x, however, there is a significant difference in the P_c with different values of sl. This difference decreases as x increases.

5. Related work

WSNs are a subclass of MANETS, and much work (e.g., [30–36]) has been proposed for securing MANETS in general. These studies are not applicable to WSNs because they assume laptop- or palmtop-level resources, which are orders of magnitude larger than those available in WSNs. Conventional public key based solutions are such an example.

Among the studies specifically targeted to resource-constrained WSNs, some [3,4] have focused on attacks and vulnerabilities. Wood and Stankovic [4] surveyed a number of denial of service attacks against WSNs, and discussed some possible countermeasures. Karlof and Wagner [3] focused on routing layer attacks, and showed how some of the existing WSN protocols are vulnerable to these attacks.

Of those offering cryptographic solutions, a reasonable number (e.g., [5–20,22,37]) have focused on efficient key management of symmetric schemes without tying them to a particular network organization. Others, recently, have been investigating more efficient techniques of public key cryptography. By using elliptic curve cryptography [38,39], for example, it has been shown (e.g., [40–42]) that sensor nodes are indeed able to compute public key operations. However, public key authentication in the context of WSNs is still an open problem, as they cannot afford a conventional public key infrastructure and the proposed alternatives (e.g., [43]) are not applicable to all contexts.

Perrig et al. [27] proposed SPINS. SPINS includes two efficient symmetric key based security building blocks: SNEP and μTESLA. SNEP provides confidentiality, authentication, and freshness between nodes and the BS, and μTESLA provides authenticated broadcast. μTESLA, which we use in our solution, implements the asymmetry required for authenticated broadcast using one-way key chains constructed with cryptographically secure hash functions, and delayed key disclosure.

Hierarchical WSNs have quite particular organization patterns, and one can take them into account to design tailored solutions. Carman et al. [5] have suggested using higher powered nodes for key generation and management functions, but did not offer concrete protocols. Kong et al. [44] and Bohge and Trappe [45] devised solutions for concrete hierarchical and heterogeneous networks. They both assume more powerful nodes, and use public key cryptography. More specifically, the former relies on RSA certificates to guarantee authentication. The amount of computation and space resources required by RSA certificates makes this solution infeasible in our context. In addition, it proposes *end-to-end* transport layer security, which prevents data aggregation at intermediary hops. The latter proposes an authentication framework for a concrete 2-tier network organization, in which a middle tier of more powerful nodes were introduced between the BS and the ordinary sensors to carry out authentication functions. However, except for the lowest tier nodes, all other nodes perform public key operations. Finally, Oliveira et al. [54] have proposed a solution for hierarchical WSNs that relies exclusively on symmetric key schemes. However, the solution is not adequate for networks where clusters are formed dynamically and periodically.

There has also been some work on detecting misbehaving nodes. E.g., Marti et al. [46] proposed a watchdog scheme that enables network nodes to detect selective forwarding attacks staged by their next hop neighbors.

Detecting and dealing with bogus data have also been focus of research. Zhu et al. [47] proposed an interleaved hop-by-hop authentication scheme to prevent injection of false data into sensor networks. The proposal makes sure that the BS can detect a false report when no more than a certain number t of nodes are compromised. Yea et al. [37] proposed SEF, a statistical en-route filtering mechanism for detecting and dropping bogus reports while being forwarded. It allows both the BS and the en-route nodes to detect false data with a certain probability. Przydatek et al. [48] proposed SIA, a framework for secure information aggregation in WSNs which makes use of random sampling strategies for allowing an user to infer about the legitimacy of a value.

Other efforts have focused on more specific types of attacks. Hu et al. [49] studied and offer solutions for wormhole attacks, whereas Newsome et al. [50] investigated sybil attacks in the context of WSNs. Finally, Deng [51] et al. address secure in-network processing, and propose a collection of mechanisms for delegating trust to aggregators that a priori are not trusted by common sensors. The mechanisms address both dissemination and aggregation of data.

6. Conclusion

In this paper we presented SecLEACH, a protocol for securing LEACH-based networks. SecLEACH achieves baseline security by adapting random key predistribution and μTESLA, and can yield different performance numbers on efficiency and security depending on its various parameter values. Our estimates show that the overhead incurred by SecLEACH is manageable; and memory usage, energy efficiency, and security level can be each traded off for another, depending on what is most critical in a system. Finally, SecLEACH preserves the structure of the original LEACH, including its ability to carry out data aggregation.

Appendix A. Average distance estimates

Next, we will present the derivation for the distance estimates. It is worth noting that both are independent of the network size n.

A.1. Distance between CH and BS

Given a square of side length $2s$ the probability P that the distance of a randomly chosen point in the square to its center is less or equal to x is given by

$$P(d\leqslant x) = \frac{\pi x^2}{4s^2} \quad \text{if } 0\leqslant x\leqslant s$$
$$= \frac{\pi x^2 - 4\left(x^2 \arctan\left(\frac{\sqrt{x^2-s^2}}{s}\right) - s\sqrt{x^2-s^2}\right)}{4s^2} \quad \text{if } s\leqslant x\leqslant \sqrt{2}s.$$

Hence, this probability density function (pdf) has the form:

$$f(x) = \frac{\pi x}{2s^2} \quad \text{if } 0\leqslant x\leqslant s$$
$$= \frac{\pi x - 4x \arctan\left(\frac{\sqrt{x^2-s^2}}{s}\right)}{2s^2} \quad \text{if } s\leqslant x\leqslant \sqrt{2}s.$$

Now, we may use the pdf to calculate the expected distance:

$$E(X) = \int_0^{\sqrt{2}s} xf(x) = \frac{(\sqrt{2}+\log(1+\sqrt{2}))s}{3}.$$

A.2. Distance between ordinary node and CH

In a population of i individuals distributed at random in an area a, the expected distance from an individual to its nearest neighbor (NN) [52], and the same with edge effect correction [53] (NN_c), are, respectively, given by

$$\mathrm{NN} = 0.5\sqrt{\frac{1}{\rho}},$$

$$\mathrm{NN_c} = 0.5\sqrt{\frac{a}{i}} + \left(0.0514 + \frac{0.041}{\sqrt{i}}\right)\frac{p}{i},$$

where ρ stands for neighborhood density, i.e., $\rho = i/a$ and p is the perimeter of the study area a.[3]

To determine the expected distance from an ordinary node to its nearest CH, we may consider only the CHs as neighbors of this node and apply the formula for NN calculation. Thus, this expected distance for LEACH and SecLEACH are given,

[3]Individuals have been treated as dimensionless points by Clark and Evans, since their dimensions are usually negligible as compared to the total area.

respectively, by

$$\mathrm{NN}_{\mathrm{LEACH}} = 0.5\sqrt{\frac{a}{h+1}} + \frac{\left(0.0514 + \frac{0.041}{\sqrt{h+1}}\right)p}{h+1},$$

$$\mathrm{NN}_{\mathrm{SecLEACH}} = 0.5\sqrt{\frac{a}{h_{\mathrm{e}}+1}} + \frac{\left(0.0514 + \frac{0.041}{\sqrt{h_{\mathrm{e}}+1}}\right)p}{h_{\mathrm{e}}+1},$$

where h and h_{e}, as defined in Section 4.1, stand for the number of accessible CHs by an ordinary node in LEACH and SecLEACH, respectively.

References

[1] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, Next century challenges: scalable coordination in sensor networks, in: Mobile Computing and Networking (MobiCom'99), Seattle, WA, 1999, pp. 263–270.

[2] G.J. Pottie, W.J. Kaiser, Wireless integrated network sensors, Comm. ACM 43 (5) (2000) 51–58.

[3] C. Karlof, D. Wagner, Secure routing in wireless sensor networks: attacks and countermeasures, Elsevier's AdHoc Networks J. 1(2–3): 293–315, 2003 (special issue on sensor network applications and protocols) Also appeared in 1st IEEE International Workshop on Sensor Network Protocols and Applications.

[4] A.D. Wood, J.A. Stankovic, Denial of service in sensor networks, IEEE Comput. 35 (10) (October 2002) 54–62.

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

[6] L. Eschenauer, V.D. Gligor, A key management scheme for distributed sensor networks, in: 9th ACM Conference on Computer and Communications Security (CCS'02), ACM New York, 2002, pp. 41–47.

[7] H. Chan, A. Perrig, D. Song, Random key predistribution schemes for sensor networks, in: IEEE Symposium on Security and Privacy (S&P'03), May 2003, pp. 197–213.

[8] S. Zhu, S. Setia, S. Jajodia, LEAP: efficient security mechanisms for large-scale distributed sensor networks, in: 10th ACM Conference on Computer and Communication Security, ACM Press, New York, 2003, pp. 62–72.

[9] D. Liu, P. Ning, Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks, in: 10th Annual Network and Distributed Systems Security Symposium (NDSS'03), 2003, pp. 263–276.

[10] D. Liu, P. Ning, Location-based pairwise key establishments for static sensor networks, in: 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'03), ACM Press, New York, 2003, pp. 72–82.

[11] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, J. Zhang, Fast authenticated key establishment protocols for self-organizing sensor networks, in: 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA'03), ACM Press, New York, 2003, pp. 141–150.

[12] S. Zhu, S. Xu, S. Setia, S. Jajodia, Establishing pairwise keys for secure communication in ad hoc networks: a probabilistic approach, in: 11th IEEE International Conference on Network Protocols (ICNP'03), Atlanta, November 2003, pp. 326–335.

[13] R. Di Pietro, L.V. Mancini, A. Mei, Random key-assignment for secure wireless sensor networks, in: SASN '03: of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks, New York, USA, 2003, pp. 62–71.

[14] W. Du, J. Deng, Y.S.H.S. Chen, P. Varshney, A key management scheme for wireless sensor networks using deployment knowledge, in: Conference of the IEEE Communications Society (INFOCOM'04), 2004.

[15] J. Hwang, Y. Kim, Revisiting random key pre-distribution schemes for wireless sensor networks, in: 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks, ACM Press, New York, 2004, pp. 43–52.

[16] D. Huang, M. Mehta, D. Medhi, L. Harn, Location-aware key management scheme for wireless sensor networks, in: 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04), ACM Press, New York, 2004, pp. 29–42.

[17] R. Kannan, L. Ray, A. Durresi, Efficient key pre-distribution schemes for sensor networks, in: 1st European Workshop on Security in Wireless and Ad-Hoc Sensor Networks (ESAS' 04), Heidelberg, Germany, August 2004.

[18] S.A. Çamtepe, B. Yener, Combinatorial design of key distribution mechanisms for wireless sensor networks. in: 9th European Symposium on Research Computer Security (ESORCS' 04), Lecture Notes in Computer Science, Sophia Antipolis, France, September 2004, pp. 293–308.

[19] D. Liu, P. Ning, R. Li, Establishing pairwise keys in distributed sensor networks, ACM Trans. Inf. Syst. Security, 8 (1) (2005) 41–77, 2005. Also appeared in 10th ACM CCS '03.

[20] W. Du, J. Deng, Y.S. Han, P.K. Varshney, J. Katz, A. Khalili, A pairwise key pre-distribution scheme for wireless sensor networks, ACM Trans. Inf. Syst. Security 8 (2) (2005) 228–258.

[21] H. Chan, A. Perrig, PIKE: peer intermediaries for key establishment in sensor networks, in: The 24th Conference of the IEEE Communications Society (INFOCOM'05), March 2005.

[22] R. Di Pietro, L.V. Mancini, A. Mei, Efficient and resilient key discovery based on pseudo-random key pre-deployment, Wireless Networks 12(6) (2006). Also appeared in 4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks 2004.

[23] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor network, IEEE Commun. Mag. 40 (8) (August 2002) 102–114.

[24] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, in: IEEE Hawaii International Conference on System Sciences, January 2000, pp. 4–7.

[25] O. Younis, S. Fahmy, Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach, in: Conference of the IEEE Communications Society (INFOCOM'04), 2004, pp. 629–640.

[26] B. Schneier, Applied Cryptography, second ed., Wiley, New York, 1996.

[27] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, SPINS: security protocols for sensor networks, Wireless

Networks 8 (5) (September 2002) 521–534, also appeared in MobiCom'01.

[28] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, System architecture directions for networked sensors, in: 9th International Conference on Architectural Support for Programming Languages and Operating Systems, 2000, pp. 93–104.

[29] C. Karlof, N. Sastry, D. Wagner, Tinysec: a link layer security architecture for wireless sensor networks, in: 2nd ACM SensSys, November 2004, pp. 162–175.

[30] L. Zhou, Z.J. Haas, Securing ad hoc networks, IEEE Network 13 (6) (1999) 24–30.

[31] S. Capkun, L. Buttyan, J.P. Hubaux, Self-organized public-key management for mobile ad hoc networks, IEEE Trans. Mobile Comput. 2 (1) (January–March 2003) 17.

[32] J.-P. Hubaux, L. Buttyán, S. Capkun, The quest for security in mobile ad hoc networks, in: 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing, ACM Press, New York, 2001, pp. 146–155.

[33] S. Capkun, J.-P. Hubaux, Biss: building secure routing out of an incomplete set of security associations, in: ACM Workshop on Wireless Security (WISE'03), ACM Press, New York, 2003, pp. 21–29.

[34] L. Venkatraman. D.P. Agrawal, A novel authentication scheme for ad hoc networks, in: IEEE Wireless Communications and Networking Conference, 2002, pp. 1268–1273.

[35] Y.-C. Hu, A. Perrig, D.B. Johnson, Ariadne: a secure on-demand routing protocol for ad hoc networks, in: 8th Annual International Conference on Mobile Computing and Networking, ACM Press, New York, 2002, pp. 12–23.

[36] Y. Zhang, W. Lee, Intrusion detection in wireless ad-hoc networks, in: 6th Annual International Conference on Mobile Computing and Networking, ACM Press, New York, 2000, pp. 275–283.

[37] F. Yea, H. Luo, S. Lu, L. Zhang, Statistical en-route filtering of injected false data in sensor networks, in: Conference of the IEEE Communications Society (INFOCOM'04), 2004.

[38] V. Miller, Uses of elliptic curves in cryptography, advances in cryptology, in: Crypto'85, Lecture Notes in Computer Science, vol. 218, Berlin, Springer, 1986, pp. 417–426.

[39] N. Koblitz, Elliptic curve cryptosystems, Math. Comput. 48 (1987) 203–209.

[40] N. Gura, A. Patel, A. Wander, H. Eberle, S.C. Shantz, Comparing elliptic curve cryptography and rsa on 8-bit cpus, in: Workshop on Cryptographic Hardware and Embedded Systems (CHES), Cambridge, MA, USA, 2004, pp. 119–132.

[41] D.J. Malan, M. Welsh, M.D. Smith, A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography, in: 1st IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON'04), Santa Clara, California, October 2004.

[42] E.-O. Blaß, M. Zitterbart, Towards acceptable public-key encryption in sensor networks, in: The 2nd International Workshop on Ubiquitous Computing, ACM SIGMIS, May 2005.

[43] W. Du, R. Wang, P. Ning, An efficient scheme for authenticating public keys in sensor networks, in: 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '05), New York, NY, USA, ACM Press, New York, 2005, pp. 58–67.

[44] J. Kong, H. Luo, K. Xu, D.L. Gu, M. Gerla, S. Lu, Adaptive Security for Multi-layer Ad-hoc Networks, Wiley New York, Wireless Commun. Mobile Comput. 2 (5) (2002) 533–547 (special issue).

[45] M. Bohge, W. Trappe, An authentication framework for hierarchical ad hoc sensor networks, in: 2003 ACM Workshop on Wireless Security, 2003, pp. 79–87.

[46] S. Marti, T.J. Giuli, K. Lai, M. Baker, Mitigating routing misbehavior in mobile ad hoc networks, in: Mobile Computing and Networking, 2000, pp. 255–265.

[47] S. Zhu, S. Setia, S. Jajodia, An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks, in: IEEE Symposium on Security and Privacy, 2004. pp. 259–271.

[48] B. Przydatek, D. Song, A. Perrig, SIA: secure information aggregation in sensor networks, in: ACM SenSys 2003, November 2003.

[49] Y.-C. Hu, A. Perrig, D.B. Johnson, Packet leashes: a defense against wormhole attacks in wireless sensor networks, in: Conference of the IEEE Communications Society (INFOCOM'03), 2003.

[50] J. Newsome, R. Shi, D. Song, A. Perrig, The sybil attack in sensor networks: analysis and defenses, in: IEEE International Conference on Information Processing in Sensor Networks (IPSN 2004), April 2004.

[51] J. Deng, R. Han, S. Mishra, Security support for in-network processing in wireless sensor networks, in: 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'03), ACM Press, New York, 2003, pp. 83–93.

[52] P.J. Clark, F.C. Evans, Distance to nearest neighbor as a measure of spatial relationships in populations, Ecology 35 (4) (1954) 445–453.

[53] K.P. Donnelly, Simulations to determine the variance and edge effect of total nearest neighbour distance, in: I. Hodder (Ed.) Simulation Studies in Archaeology, 1978.

[54] L.B. Oliveira, H.C. Wong, A.A.F. Loureiro, LHA-SP: Secure protocols for hierarchical wireless sensor networks, in: Ninth International Symposium on Integrated Network Management (IM'05), IFIP/IEEE, May 2005, Nice, France, pp. 31–44.