J Supercomput (2008) 45: 44–65
DOI 10.1007/s11227-008-0184-2

Scalable and efficient key management for heterogeneous sensor networks

Firdous Kausar · Sajid Hussain · Laurence T. Yang · Ashraf Masood

Published online: 23 February 2008

Abstract As typical wireless sensor networks (WSNs) have resource limitations, predistribution of secret keys is possibly the most practical approach for secure network communications. In this paper, we propose a key management scheme based on random key predistribution for heterogeneous wireless sensor networks (HSNs). As large-scale homogeneous networks suffer from high costs of communication, computation, and storage requirements, the HSNs are preferred because they provide better performance and security solutions for scalable applications in dynamic environments. We consider hierarchical HSN consisting of a small number high-end sensors and a large number of low-end sensors. To address storage overhead problem in the constraint sensor nodes, we incorporate a key generation process, where instead of generating a large pool of random keys, a key pool is represented by a small number of generation keys. For a given generation key and a publicly known seed value, a keyed-hash function generates a key chain; these key chains collectively make a key pool. As dynamic network topology is native to WSNs, the proposed scheme allows

F. Kausar · A. Masood
College of Signals, National University of Science and Technology, Rawalpindi, Pakistan

F. Kausar
e-mail: firdous.imam@gmail.com

A. Masood
e-mail: ashrafm61@gmail.com

F. Kausar · S. Hussain (✉)
Jodrey School of Computer Science, Acadia University, Wolfville, NS, B4P 2R6, Canada
e-mail: sajid.hussain@acadiau.ca
url: http://cs.acadiau.ca/~shussain

L.T. Yang
Department of Computer Science, St. Francis Xavier University, Antigonish, NS, B2G 2W5, Canada
e-mail: ltyang@stfx.ca
url: http://cse.stfx.ca/~ltyang/

dynamic addition and removal of nodes. This paper also reports the implementation and the performance of the proposed scheme on Crossbow's MicaZ motes running TinyOS. The results indicate that the proposed scheme can be applied efficiently in resource-constrained sensor networks. We evaluate the computation and storage costs of two keyed-hash algorithms for key chain generation, HMAC-SHA1 and HMAC-MD5.

Keywords Security · Key management · Heterogeneous sensor networks · Random key predistribution · Authentication

1 Introduction

Wireless sensor networks (WSNs) are commonly used in ubiquitous and pervasive applications such as military, homeland security, health-care, and industry automation. WSNs consist of numerous small, low-cost, independent sensor nodes, which have limited computing and energy resources. Secure and scalable WSN applications require efficient key distribution and key management mechanisms. Cryptography is the foundational technology used for protecting and securing the communication in sensor networks [17]. This technology relies on keys as the centerpieces, and many attacks focus on disclosing these keys. As a result, the management of the keys (the process by which keys are generated, stored, protected, distributed, used, and destroyed) is a very important and challenging problem in a large-scale network consisting of several hundreds or thousands of sensor nodes.

Conventional security protocols are usually master key based or distributed key based management schemes. In master key based schemes, every node shares a single preloaded master key. Further, master key is used to negotiate session keys for securing different wireless links. For example, Menezes et al. [25] use a simple three-way handshaking and authentication protocol based on the master key for setting up session keys. This type of key management scheme has the underlying assumption that the sensor nodes are tamper proof and the master key that is stored inside each node cannot be retrieved by the adversary. However, the assumption that the nodes are tamper proof cannot be ensured in many sensor network applications because sensor nodes are usually left unattended in hostile and remote environments. Once the master key is compromised, the adversary can use it to break the security of the entire network.

Other commonly used schemes in WSNs are key predistribution schemes [5, 6, 8, 9, 14, 26, 28, 33]. In these approaches, with minimal resources, one can achieve a known probability of connectivity within a network. These efforts assume a deployment of homogeneous nodes and, therefore, use a balance distribution of random keys among the nodes. Most existing research mainly considers homogeneous sensor networks, where all sensor nodes have identical capabilities in terms of communication, computation, sensing, and reliability; however, homogeneous WSNs are not scalable.

Several recent works, on the other hand, investigate heterogeneous sensor networks (HSNs). Girod et al. [16] develop tools to support heterogeneous systems as well as the measurement and visualization of operational systems. Lazos and Poovendran [20] study the coverage problem in planar heterogeneous sensor networks and

formulate the coverage problem as a set intersection problem. They formulate expressions in order to determine the required number of sensors for a field of interest. Ma et al. [24] propose a resource oriented protocol for heterogeneous sensor networks to build the network model that adapts according to the members' resources. Du and Lin [10] propose a differentiated coverage algorithm which can provide different coverage degrees for different areas; the algorithm is energy efficient since it only keeps minimum number of sensors in active state. Duarte-Melo and Liu [13] analyze the energy consumption and lifetime of HSN by providing periodic data from a sensing field to a remote receiver.

In this paper, we propose a scalable protocol for key management that is sensitive to the sensor nodes resource constraints, including storage, computation and communication. The proposed key management scheme is based on random key predistribution for HSNs; the contributions of the proposed scheme are as follows:

- We consider heterogeneous sensor network consisting of two types of sensors: high-end (H-sensor) and low-end (L-sensor). Further, for scalable solutions, the proposed scheme uses hierarchical structure, where H-sensor act as cluster head (CH) and L-sensors as cluster members.
- We propose an efficient method for key management that uses a keyed-hash-chain based technique for keys generation. Instead of generating a large pool of random keys, a key pool is represented by a small number of generation keys. For a given generation key and a publicly known seed value, a keyed-hash function generates a key chain; these key chains collectively make a key pool. Further, each sensor node is assigned a small number of randomly selected generation keys. As a result, by using generation keys, the proposed scheme significantly reduces the storage requirements.
- Dynamic network topology is native to WSNs because nodes can fail or be added. As result, the proposed scheme allows dynamic node addition and removal. In the case of node addition, the proposed scheme is able to distinguish between legitimate and malicious nodes. Further, as adversaries can compromise sensors and acquire all security information, a rekeying scheme is incorporated to update all types of keys periodically.
- We implement our scheme in a real sensor network, where we consider the computation and storage costs of two keyed-hash algorithms for key chain generation, HMAC-SHA1 and HMAC-MD5.

The rest of paper is organized as follows: Section 2 provides the related work, Sect. 3 describes the network and threat models, Sect. 4 describes the proposed scheme in detail, Sect. 5 gives the results and performance evaluation, Sect. 6 provides the TinyOS implementation details, and Sect. 7 concludes the paper.

2 Related work

The key management problem is a very active research area in WSNs. Two classical solutions to this problem are: (1) public-key cryptography (Deffie Hellman) and (2) schemes based on a trusted server (Kerberos). The first solution is infeasible

for sensor network because public key cryptography such as RSA or Elliptic Curve cryptography (ECC) is unsuitable for most sensor architectures due to high energy consumption and increased code storage requirements. The second solution is also not applicable in sensor networks since most of the sensors in the field are unable to directly communicate with the sink due to their very short communication range and the wide-spread distribution of the sensors. Hence, sensor networks cannot rely on a trusted server for key distribution. Several alternative approaches have been developed to perform key management on resource-constrained sensor networks which includes random key predistribution schemes, plaintext key exchange schemes, and transitory master key schemes.

Eschenauer and Gligor [14] propose a probabilistic key predistribution technique to bootstrap the initial trust between sensor nodes. Generate a large pool of random symmetric keys and then preconfigured each node with a number of keys randomly selected from the key pool without replacement. Neighboring nodes use their preconfigured keys to set up their pairwise keys. A communication channel secured between two nodes using pairwise keys is called a key path. To protect confidentiality, every key is usually assigned an index, and during shared key discovery, nodes exchange the index of keys with neighbors to ultimately determine their shared pairwise keys. 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 key path at the end of shared key discovery. If the network density, the size of the key pool, and the number of keys preconfigured in each sensor node are carefully chosen, it is highly likely that all nodes in the network will be connected via key paths.

Chan et al. [5] propose the q-composite key predistribution, which allows two sensors to setup a pairwise key only when they share at least q common keys. It is shown in [5] that the q-composite scheme can achieve greatly strengthened security under a small scale attack while trading off increased vulnerability in the face of a large scale physical attack on network nodes. In addition, they also propose scheme allowing keys to be distributed to nodes in pairs so that keys may be associated with specific nodes, thus allowing authentication. Chan and Perrig [4] also develop a protocol named PIKE for key establishment by using peer sensor nodes as trusted intermediaries.

Some location-aware schemes which improve the security of the key predistribution schemes are proposed in [21, 31]. These techniques divide the target field into nonoverlapping square areas and randomly deploy the sensors in every area. The exact location of a sensor in any area is unknown, but there is knowledge about the identity of sensors in every area. This information helps to eliminate the dependency of keys between nonadjacent cells. The idea of threshold key predistribution schemes is proposed in [1] and further studied in [2].

In one of the variations of such key predistribution schemes, every sensor stores the coefficients of a symmetric bivariate polynomial that is evaluated at one of its variables. Liu et al. [22] propose two key distribution schemes based on bivariate polynomials. They propose the random subset assignment scheme and the hypercube-based key predistribution scheme in which the key pool consists of multiple instances of polynomial. Oliveira et al. [26] use random key predistribution for secure communication in hierarchical (cluster-based) protocols such as LEACH [18]. These and

some others [6, 8, 9, 28, 33] have assumed a deployment of homogeneous nodes, and have, therefore, suggested a balanced distribution of random keys to each of the nodes to achieve security. Most of these schemes suffer from high communication and computation overhead and/or high storage requirement.

There is also active research in key management for heterogeneous sensor networks. Du et al. [12] propose the asymmetric predistribution (AP) scheme for heterogeneous sensor networks which provides better security with low complexity and significant reduction on storage requirement by storing more numbers of preconfigured keys on high end sensors and a small number of pre-configured keys on low end sensors. Liu et al. [23] propose a framework for key management schemes in distributed wireless sensor networks with heterogeneous sensor nodes. Traynor et al. [29] demonstrate that a probabilistic unbalanced distribution of keys throughout the network that leverages the existence of a small percentage of more capable sensor nodes cannot only provide an equal level of security, but also reduce the consequences of node compromise.

Bulusu et al. [3] propose two key predistribution based scheme for heterogeneous networks which consist of nodes which are stationary as well as highly mobile. They use a separate disjoint key pool to establish links between the stationary and mobile nodes of the network because if the same key pool is used in multiple networks, the compromise of keys in one network would lead to compromise of keys in all the networks. Traynor et al. [30] characterize the effects of the unbalanced key management system, and design a complementary suite of key establishment protocols known as LIGER. Using their predeployed keys, nodes operating in isolation from external networks can securely and efficiently establish keys with each other.

3 Network model

We consider the heterogeneous sensor network consisting of three types of nodes: base station, H-sensor, and L-sensor, as described below:

- *Base Station*: The base station is assumed to be secure, not prone to failures, and does not have any resource constraints such as bandwidth, energy, memory, and processing.
- *H-Sensors*: H-sensors have more memory and processing capability. These nodes are equipped with tamper resistant hardware and communicate directly with the base station. Although H-sensors have rich resources, but these are still limited as compared to the base station. For instance, Crossbow's stargate nodes can be used as H-sensors.
- *L-Sensors*: L-sensors are ordinary sensor nodes that are limited in terms of memory and processing capability. The L-sensors acquire data from the surrounding environment and forward the collected data to the H-sensor.

In a typical initial HSN deployment, there would be a small number of H sensors and a large number of L-sensors. Further, for scalability and easy maintenance, both H-sensors and L-sensors can be added as needed. H-sensors and L-sensors are assumed to be uniformly and randomly distributed in the field. Clustering of sensors

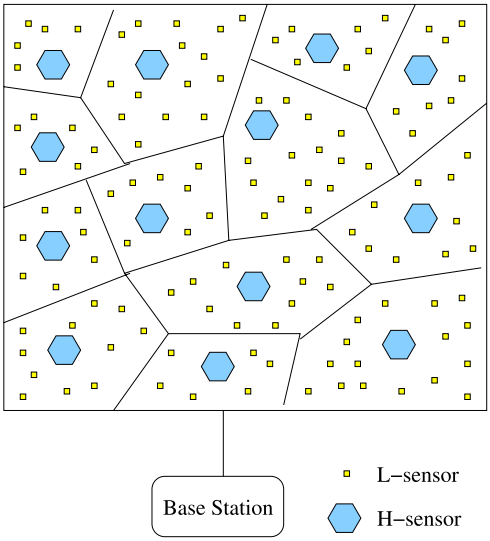

Fig. 1 Network model

enable local data processing, which reduces communication load in the network in order to provide scalable solutions.

HSN consists of a hierarchical structure where sensors are divided into clusters and each cluster is managed by a cluster head, as shown in Fig. 1. All H-sensors act as cluster heads; whereas each L-sensor is a cluster member and cannot act as a cluster head. Each L-sensor is able to securely communicate with all other L-sensors in its neighborhood and its cluster head (H-senor). Moreover, H-sensors maintain secure communication with following entities: base station, cluster member (L-sensor) and other cluster heads (H-sensors).

3.1 Threat model

A malicious node can be either an external node that does not know the cryptographic keys, or an internal node (L-sensor), that possesses the keys. An adversary can create an internal compromised node by capturing a legitimate L-sensor node. All these malicious nodes can exhibit Byzantine behavior which can be described as a behavior when one or more sensors or devices work in collusion to disrupt the network. It could include several security challenges such as denial of service attack, dropping or altering packets, topology distortion, impersonation, and wormholes.

4 Proposed scheme

In this section, we present an efficient key management scheme designed for heterogeneous sensor networks. The proposed scheme uses a symmetric-key mechanism to distribute, revoke, and renew keys during the lifetime of HSN. A few terms and definitions used in the rest of the paper are as follows:

Table 1 Notations

Notation	Definition
BS	Base station
CH	Cluster head
K	A pool of keys
M	Total number of key chains
C_i	Key chain i
gk_i	Generation key of i-th key chain
R_X	Set of the keys in node X's key ring
r	A number of keys in L-sensor key ring
S	A number of keys in H-sensor key ring
K_M	Master key
K_C	Cluster broadcast key
K_{M,L_i}	Authentication key of L-sensor i
$K_{X,Y}$	A shared key between X and Y
id_{L_i}	Identity of L-sensor i
$E_K(m)$	An encryption of message m with key K
$nonce_{L_i}$	A random number string generated by L-sensor i
$MAC_K(A\|\|B)$	MAC calculated using key K on message A and B

Definition 1 A one way hash function is a function H satisfying the following conditions:

- The argument x can be of arbitrary length and the result $H(x, k)$ has fixed length of n bits.
- The hash function must be one way in the sense that given a y in the image of H, it is computationally infeasible to find a message x such that $H(x) = y$, and given x and $H(x)$, it is computationally infeasible to find a message $\acute{x} \neq x$ such that $H(\acute{x}) = H(x)$.

Terms

- *Key Pool*: A key pool K is a large pool of random symmetric keys.
- *Key Chain*: A key chain C is a subset of K, $C \subseteq K$. Each key chain is generated independently via a unique generation key and a publicly known seed S by applying a keyed hash algorithm repeatedly. A publicly known seed value is same for every key chain. Each key chain can be uniquely identified as C_i, where $i = 0 \ldots M - 1$. The key pool K consists of M equal sized key chains as given below:

$$K = C_0 \cup C_1 \cdots \cup C_{M-1}. \tag{1}$$

- *Key Ring*: A key ring R consists of randomly selected generation keys of corresponding key chains. Each sensor node is assigned a ring of R keys.

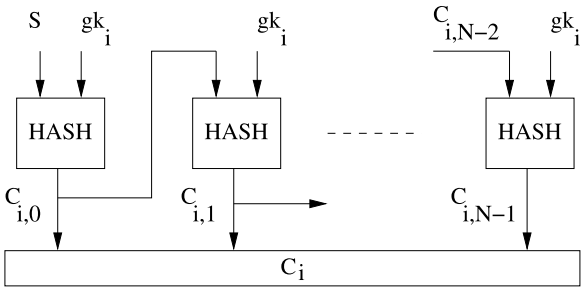

Fig. 2 Key chain generation process

4.1 Key predistribution phase

The key predistribution phase includes key pool generation and key ring assignment.

4.1.1 Key pool generation

In this section, we describe the process of key pool generation. Generate a large pool of random symmetric keys as follows. First, the cardinality (size) of key pool $|K|$ is selected. Then the number of key chains M is chosen accordingly. A key pool K consists of M different key chains, as given in (1). Further, there are no common keys between any two key chains, which is formally given as follows:

$$C_i \cap C_j = \phi, \quad \forall i \neq j. \tag{2}$$

As a key chain C_i is generated independently via a unique generation key gk_i and publicly known seed S by applying a keyed hash algorithm repeatedly [28], the j-th key of the key chain C_i is computed as:

$$k_{C_{i,j}} = \begin{cases} H(gk_i, S): & j = 0, \\ H(gk_i, k_{C_{i,j-1}}): & 1 \leq j \leq N-1. \end{cases} \tag{3}$$

Figure 2 shows a block diagram to illustrate the process of key generation. The first key is generated by using seed S and gk_i as inputs to Hash (H); however, the remaining keys are generated by applying H over gk_i and the previous key. The total number of keys in a key chain is N, where

$$N = \frac{K}{M}. \tag{4}$$

Further, base station generates a special key K_M known as master key, which is used for authentication.

4.1.2 Key ring assignment

For each node (L-sensor or H-sensor), a unique identity (id) is generated using a pseudo-random function (PRF). Before deploying the nodes, each node is loaded with its assigned key ring R, where R consists of the number of generation keys used to generate corresponding key chains. The assigning rules are as follows:

- Each L-sensor node is assigned r number of randomly selected generation keys of corresponding key chains. First, input the L-sensor id as seed to pseudo-random number generator (PRNG) of a large enough period to produce a sequence of r numbers, as given below: $PRNG(id_i) = n_1, n_2, \ldots, n_r$. Second, the set of key ids assigned to L-sensor can then be obtained by mapping each number in the sequence to its correspondent value modulus M, as given below:

$$id_{gk_i} = n_i \bmod M, \tag{5}$$

  where $0 \leq id_{gk_i} \leq M - 1$. From these r generation keys, $r \times N$ random keys can be calculated effectively. In addition, each L-sensor is preloaded with an authentication key K_{M,L_i}, which is generated by applying one way hash function on the id of L-sensor and master key, i.e., $K_{M,L_i} = f(K_M, id_{L_i})$.
- Each H-sensor node is preloaded with S randomly selected generation keys of corresponding key chains as described above. However, it should be noted that $S \gg r$. Each H-sensor is also preloaded with a master key K_M.

4.2 Cluster formation phase

During the cluster formation phase, all H-sensors broadcast Hello messages to nearby L-sensors with some random delay, in order to avoid collisions of Hello messages from neighboring H-sensors. The probability of collision is quite small when a non-persistent CSMA protocol is used for medium access control [32]. Moreover, an H-sensor can broadcast its ID multiple times to increase the probability that it is received by all its neighbors. The Hello message includes the ID of the H-sensor. The transmission range of the broadcast is large enough so that all L-sensors can receive Hello messages from several H-sensors. Then each L-sensor selects the H-sensor as the cluster head whose Hello message has the best received signal strength indicator (RSSI) value. Each L-sensor also records the ids of other H-sensors from which it receives the Hello messages, and these H-sensors are listed as backup cluster heads in case the primary cluster head fails. Only H-sensor can act as a cluster head (CH); whereas the L-sensors act as cluster members; the details of clustering scheme can be found in [11].

4.3 Cluster head based shared key discovery phase

The shared key discovery phase begins after cluster formation phase. First, each cluster member sends to its cluster head a message, which includes its ID, nonce, its neighboring nodes information, and MAC which is calculated on all these values using a key K_{M,L_i}.

Second, this phase also includes a neighborhood discovery, as shown in Fig. 3. In message 1, L-sensor (L_i) broadcasts hello messages for a short range in order to discover neighbors. In message 2, one of L_i's neighbor, say L_j acknowledges with *HelloReply* message. Then L_i adds L_j's id in its neighbors list. The neighborhood discovery phase ends when all the L-sensors have obtained neighborhood information.

Fig. 3 Neighboring node discovery

1: $L_i \Rightarrow *: Hello(id_{L_i})$
2: $L_j \Rightarrow L_1: HelloReply(id_{L_j})$
3: L_i: adds the id_{L_j} into *List*
4: L_j: Repeat 2 and 3 for every *HelloReply*

$$
\begin{aligned}
1: \; L_i \Rightarrow CH \quad &: \; id_{L_i}, nonce_{L_i}, List, \\
&: \; MAC_{K_{M,L_i}}(id_{L_i} \| nonce_{L_i} \| List) \\
2: \; L_j \Rightarrow CH \quad &: \; id_{L_j}, nonce_{L_j}, List, \\
&: \; MAC_{K_{M,L_j}}(id_{L_j} \| nonce_{L_j} \| List) \\
3: CH \Rightarrow \; L_i \quad &: \; n, id_{gk_m}, id_{L_i}, id_{L_j}, \\
&: \; MAC_{K_{M,L_i}}(n \| id_{gk_m} \| id_{L_i} \| id_{L_j} \| nonce_{L_i}) \\
4: CH \Rightarrow \; L_j \quad &: \; n, id_{gk_m}, id_{L_i}, id_{L_j}, \\
&: \; MAC_{K_{M,L_j}}(n \| id_{gk_m} \| id_{L_i} \| id_{L_j} \| nonce_{L_j})
\end{aligned}
$$

Fig. 4 Neighboring L-sensors with common preloaded generation key

Fig. 5 Session key generation

```
1: function GetKey (n, gk_m)
   /* param n: random number */
   /* param gk_m: common generation key */
2: K_{C_{m,0}} = H(gk_m, seed)
3: for i=1 to n-1  do
4:    K_{C_{m,i}} = H(gk_m, K_{C_{m,i-1}})
5: end for
6: return K_{C_{m,n-1}}
```

Third, CH discovers the shared generation keys between neighboring L-sensors in its cluster, as shown in Fig. 4. In messages 1 and 2, L-sensors L_i and L_j send messages to CH, where messages contain their ids, nonce, the list of their neighboring L-sensors ids, and MAC on all these values. CH determines the generation keys in L_i and L_j's key rings (R_i and R_j) by using the pseudo-random scheme described above in Sect. 4.1. The CH chooses the common generation key gk_m, where $gk_m \in R_i \cap R_j$; a generation key with minimum index is selected in case of multiple common keys. Then CH determines the shared pairwise key between L_i and L_j by generating a random number n, where $[0 \leq n \leq N-1]$, which is used as an index in the key chain C_m for selecting the pairwise shared key, i.e., $K_{C_{m,n}}$. Then CH disseminates the shared-key information to L_i and L_j sensors, as shown in messages 3 and 4. The shared-key information consists of the following: (a) ids of neighboring L-sensors (L_i and L_j), (b) id of the common generation key, i.e., id_{gk_m}, (c) n that represents the index of shared pairwise key of C_m key chain, (d) nonce, and (e) MAC that is calculated on all these values using corresponding authentication keys K_{M,L_i} and K_{M,L_j}. In other words, L-sensors L_i and L_j share n-th key of C_m key chain by applying the common key generation algorithm shown in Fig. 5 on n and gk_m. Further, L-sensors L_i and L_j also share the same n-th key of C_m key chain with their CH.

Fig. 6 Neighboring L-sensors without common preloaded generation key

$$
\begin{aligned}
1: L_x \Rightarrow CH &: id_{L_x}, nonce_{L_x}, List, \\
&: MAC_{K_{M,L_x}}(id_{L_x} \| nonce_{L_x} \| List) \\
2: L_y \Rightarrow CH &: id_{L_y}, nonce_{L_y}, List, \\
&: MAC_{K_{M,L_y}}(id_{L_y} \| nonce_{L_y} \| List) \\
3: CH \Rightarrow L_x &: id_{L_x}, id_{g_i}, p, \\
&: MAC_{K_{M,L_x}}(id_{L_x} \| id_{g_i} \| p \| nonce_{L_x}) \\
4: CH \Rightarrow L_y &: id_{L_y}, id_{g_j}, q, \\
&: MAC_{K_{M,L_y}}(id_{L_y} \| id_{g_j} \| q \| nonce_{L_y}) \\
5: CH \Rightarrow L_x &: (E_{K_{CH,L_x}}(K_{L_x,L_y})) \\
6: CH \Rightarrow L_y &: (E_{K_{CH,L_y}}(K_{L_x,L_y}))
\end{aligned}
$$

4.3.1 No common preloaded generation key between L-sensors

Some L-sensors may not share any preloaded generation key with their neighbors. For each pair of L-sensors (say X and Y) that do not share any generation key, $R_x \cap R_y = \phi$, CH obtains a shared-key between CH and L_x and a shared-key between CH and L_y. Then CH generates a pair-wise key for each pair (L_x and L_y), and securely sends the key to them.

Figure 6 shows an example for neighboring L-sensors that do not share common preloaded generation key. CH first checks if it has a preloaded generation key shared with the L-sensors (e.g., L_x and L_y), $R_x \cap R_{CH} \neq \phi$ and $R_y \cap R_{CH} \neq \phi$. As CH is preloaded with a large number of generation keys, there is a high probability that CH can find at least one shared generation key with L_x and L_y, i.e., $gk_i \in R_x \cap R_{CH}$ and $gk_j \in R_y \cap R_{CH}$. CH generates random numbers p, where $[0 \leq p \leq N-1]$ and q where $[0 \leq q \leq N-1]$. CH sends messages 3 and 4, which means that CH shares the p-th key ($K_{C_{i,p}}$) of C_i key chain with node L_x and q-th key ($K_{C_{j,q}}$) of C_j key chain with node L_y. Then CH generates a new shared key between L_x and L_y and sends this key to both L_x and L_y, encrypting with shared key between nodes (L_x, L_y) and CH, as shown in messages 5 and 6.

4.3.2 No common preloaded generation key between CH and L-sensor

In case that an L-sensor does not share any preloaded generation key with its CH, then CH finds a key to communicate securely with that L-sensor. CH generates a key K_{CH,L_i} and sends the key to L-sensor encrypted with that L-sensor's authentication key K_{M,L_i}, as given below:

$$CH \rightarrow L_i : E_{K_{M,L_i}}(K_{CH,L_i}).$$

4.4 Inter-cluster communication

An intercluster communication between CHs is achieved through a key K_{CH_i,CH_j} generated by applying hash function on id_{CH_i} and id_{CH_j} using key K_M, as given below:

$$K_{CH_i,CH_j} = H(K_M, id_{CH_i} \| id_{CH_j}).$$

$$
\begin{aligned}
1: \quad L_a \Rightarrow CH_i \quad &: \quad id_{L_a}, id_{L_b}, nonce_{L_a}, \\
&: \quad MAC_{K_{M,La}}(id_{L_a} \| id_{L_b} \| nonce_{L_a}) \\
2: CH_i \Rightarrow \quad L_a \quad &: \quad id_{L_a}, id_{L_b}, id_{gk_i}, n, \\
&: \quad MAC_{K_{M,La}}(id_{L_a} \| id_{L_b} \| id_{gk_i} \| n \| nonce_{L_a}) \\
3: CH_i \Rightarrow CH_j \quad &: \quad id_{L_a}, id_{L_b}, id_{gk_i}, n, \\
&: \quad MAC_{K_M}(id_{L_a} \| id_{L_b} \| id_{gk_i} \| n) \\
4: CH_j \Rightarrow \quad L_b \quad &: \quad id_{L_a}, id_{L_b}, id_{gk_i}, n, \\
&: \quad MAC_{K_{M,L_b}}(id_{L_a} \| id_{L_b} \| id_{gk_i} \| n)
\end{aligned}
$$

Fig. 7 L-sensor to L-sensor inter-cluster communication

If node L_a wishes to communicate with a node that lies in a different cluster, then two CHs are involved in order to setup a session key between L-sensors. Say L_a lies in Cluster C_i and L_b in C_j and the respective CHs are CH_i and CH_j. As shown in Fig. 7, in line 1, L_a sends request to CH_i consisting of its id (id_{L_a}), the id of L-sensor with which it wants to communicate (id_{L_b}), a nonce, and MAC that is calculated on all these values using K_{M,L_a}. CH_i determines the common generation key (gk_m) in the key rings of L-sensors (L_a and L_b). Then CH_i generates a random number n and sends the shared key message to L_a consisting of both L-sensors ids (id_{L_a}, id_{L_b}), id of the common generation key (id_{gk_m}), n, $nonce_{L_a}$ and MAC that is calculated by using K_{M,L_a}, as shown in line 2 of Fig. 7. Similarly, in line 3, CH_i sends a message to CH_j containing the shared key information between L_a and L_b. After receiving this message, CH_j forwards the shared key message to L_b consisting of ids of both L_a and L_b, id of the common generation key, n, and MAC that is calculated on all these values using K_{M,L_b}, as shown in line 4. Now L-sensors L_a and L_b use n-th key of C_i key chain to communicate securely.

4.5 Addition of new nodes

A desirable property in a scalable key management scheme is the ability of adding new sensors to the network. These newly deployed sensor nodes need to establish secret key with existing nodes. However, before adding new nodes into network, it should be ensured that the newly deployed sensor node is not an adversary node. The proposed scheme is robust for adding new legitimate L-sensors in the network. After an L-sensor L_x is deployed in the network, L_x determines its neighbors using neighboring node discovery as described in Fig. 3. L_x sends join request to the CH, for which it has the best RSSI and LQI values, as given below:

$$
\begin{aligned}
L_x \rightarrow CH: \quad & id_{L_x}, nonce_{L_x}, List, \\
& MAC_{K_{M,L_x}}(id_{L_x} \| nonce_{L_x} \| List).
\end{aligned}
$$

The CH authenticates the node L_x by verifying the MAC. If authenticated, CH determines the shared key for each of L_x's neighbors and unicasts the shared key message to L_x and its neighbors.

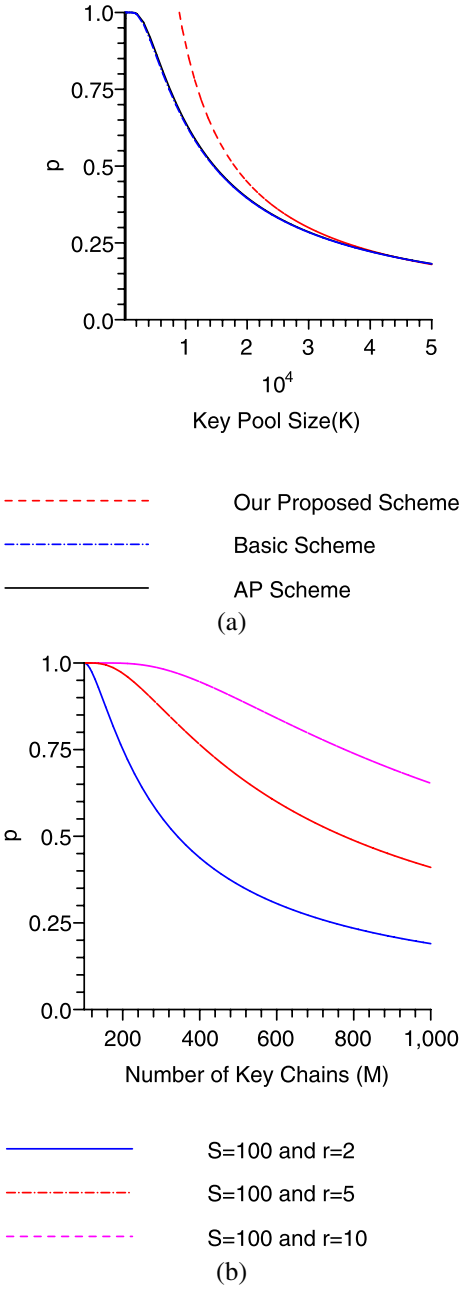

Fig. 8 The probability of key sharing

4.6 Setting up cluster key

Cluster key is used by both CH and cluster members to securely broadcast messages within a cluster. After setting up shared pairwise key between cluster members, CH

generates a cluster key K_C, which is sent to each cluster member, where K_C is encrypted with the corresponding shared key between CH and the cluster member. For example, CH sends to L_u (cluster member) the following message:

$$CH \rightarrow L_u : E_{K_{CH,L_u}}(K_C),$$

where K_{CH,L_u} is the shared key between L-sensor L_u and CH.

4.7 Key revocation

Revocation procedures are involved after detecting compromised or faulty nodes. The base station is responsible for monitoring sensor behavior and detecting a sensor failure or compromise. For a compromised node, the base station sends this information to the corresponding CH. The CH broadcasts to its member the *Revocation* message containing the list of key ids to be revoked, where the message is signed with K_C. The *Revocation* message is formed as follows:

$$list(id_{gk_1}, id_{gk_2}, \ldots id_{gk_r}), MAC_{K_C}(list).$$

Each L-sensor when receives a *Revocation* message, it verifies the MAC to check the integrity of message and to locate those key ids in its key ring, and remove the keys (if any). After key revocation, some links may disappear and affected nodes need to reconfigure those links by restarting the shared key discovery phase.

4.8 Rekeying

Using the same encryption key for extended duration may result in a cryptanalytic attack. A remedy could be to ignore this threat because it is anticipated that in most cases the lifetime of nodes would be less than the lifetime of the shared key between two nodes [27]. However, in some cases, since it is possible that lifetime of keys expires, it is necessary to renew the encryption keys (session keys). In order to accomplish the renewal of the session keys, the affected nodes remove the expired keys and restart the shared key discovery phase with CHs.

5 Performance evaluation

In this section, the proposed key distribution scheme is compared with other commonly used key distribution techniques. The results show that the proposed scheme can significantly reduce the storage requirements, while providing similar probability of key sharing among nodes.

The key pool size $\|K\|$ is a critical parameter because in random key distribution schemes 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 $\|R\|$ a fixed parameter. Once R is set then for larger values of $\|K\|$, the probability that two L-nodes will share a key is small. Further, the probability that a randomly chosen link is compromised when a node that is at neither end of the compromised link decreases by increasing the value

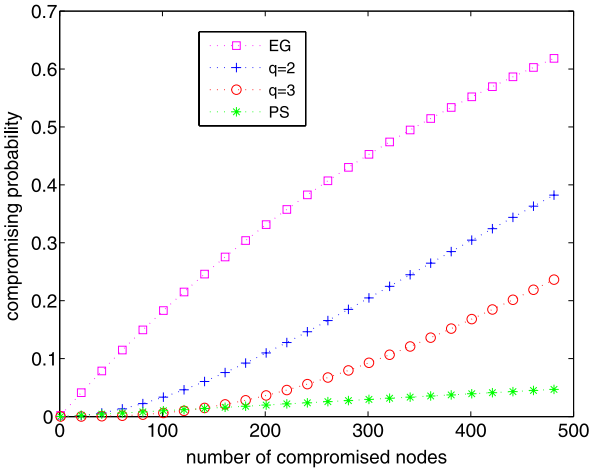

Fig. 9 The compromising probability

of $\|K\|$. We want to find the largest key pool size $\|K\|$, such that the probability of key sharing between two L-sensors, as well as L-sensor and H-sensor is not less then the threshold p.

Let p be the probability that an L-sensor and H-sensor share at least one common key in their key ring. The number of possible key ring assignments for an L-sensor is

$$\frac{M!}{r!(M-r)!}. \tag{6}$$

The number of possible key ring assignment for an H-sensor is

$$\frac{M!}{S!(M-S)!}. \tag{7}$$

The total number of possible key ring assignment for an L-sensor and H-sensor is

$$\frac{M!}{r!(M-r)!} \times \frac{M!}{S!(M-S)!}. \tag{8}$$

The probability that an L-sensor and H-sensor share a common key can be given as

$$p = 1 - \frac{(M-r)!(M-S)!}{M!(M-r-S)!}. \tag{9}$$

Figure 3 shows probability of key sharing for different schemes. For different values of K, M, S, and r, we plot the probability of sharing at least one key under our proposed scheme, the key predistribution scheme [14], which we will refer as basic scheme, and Asymmetric Predistribution scheme [12], which we will refer as AP scheme. In Fig. 8(a), the key pool size ranges from 1,000 to 50,000 and key ring size is fixed to 100 for basic scheme. For AP scheme, H-sensor keys are 500 and L-sensor keys are 20. For our proposed scheme, the number of key chains (M) varies from 100 to 1,000, $S = 90$, and $r = 2$. In other words, the number of key chains (M) is 0.02 times of the corresponding key pool size.

Figure 8(a) shows that for the proposed scheme, the same probability of key sharing among nodes can be achieved by just loading 2 generation keys in sensor node as compared to 100 keys in basic scheme [14], and 20 keys in AP scheme [12]. For instance, if there are 1,000 L-sensors and 10 H-sensors in an HSN, where each L-sensor is preloaded with 2 generation keys and each H-sensor is preloaded with 100 generation keys, the total memory requirement for our proposed scheme in the unit of key length is $2 \times 1000 + 100 \times 10 = 3{,}000$. However, in AP scheme [12], if each H-sensor is loaded with 500 keys and each L-sensor is loaded with 10 keys, the total memory requirement for storing these keys will be $500 \times 10 + 1000 \times 20 = 25{,}000$, which is 8 times larger than our proposed scheme. Further, for a homogeneous sensor network with 1,000 L-sensors, where each L-sensor is preloaded with 100 keys, the memory requirements will be $100 \times 1000 = 100{,}000$, which is 33 times larger than our proposed scheme.

Figure 8(b) shows that the probability of key sharing among nodes and CH increases by a very little increase in the number of preloaded generation keys in L-sensors. For instance, if preloaded keys are increased from 2 to 5, the key sharing probability increases from 0.5 to 0.8 approximately, for 400 key chains.

5.1 Security evaluation

In this section, we investigate the security resilience of our proposed scheme against node compromise attack. Further, we calculate the expected number of compromised links due to key revealing of captured nodes.

Each L-sensor has a knowledge of $r \times N$ keys. The probability that a given key does not belong to an L-sensor is $1 - \frac{r}{M}$. If there are n compromised nodes, the probability that a given key is not compromised is $(1 - \frac{r}{M})^n$. The probability of total number of compromised keys, where n number of L-sensors are captured, is as follows:

$$\overline{p} = 1 - \left(1 - \frac{r}{M}\right)^n. \tag{10}$$

Figure 9 shows the compromising probability with respect to the number of compromised nodes. In this figure, the proposed scheme (PS) is compared to EG [14] and q-composite [5] schemes. For the given parameters: $M = 1{,}000$, $K = 50{,}000$, $r = 5$, and $m = 100$, the results show that PS is more resilient against node capture as compared to EG and q-composite schemes.

6 Implementation in real sensor network

In this section, the implementation issues are investigated to show that the proposed scheme can be efficiently implemented on resource-constrained sensor nodes.

For L-sensors, we use Crossbow's Micaz sensor nodes; the details are as follows: 4 MHz Atmel ATmega128L processor, 128 KB of program Flash memory, 512 KB of measurement flash memory, and a 2.4 GHz ChipCon radio.

Crossbow's stargate nodes are used as H-sensors. The stargate is a gateway node with the following specifications: 400 MHz Intel PXA255 Xscale processor, 64 MB

Table 2 Time and memory requirements for MicaZ

Primitive	Time (msec)	RAM (Bytes)	ROM (Bytes)
SHA-1	10.545	128	4,048
MD5	5.757	176	12,500
HMAC-SHA1	21.959	30	4,424
HMAC-MD5	12.217	90	12,986

of SDRAM and 32 MB of flash memory. Further, on another set of tests, stargates were replaced by desktops—desktop specifications: 1.8 GHz AMD Turion(tm) 64X2 mobile TL-56 processor and 2 GB of RAM running Windows XP.

The proposed protocols are implemented for TinyOS using nesC [15] programming language. Our assessment includes how to discover the neighbors, the delay overhead of generating the shared key by applying keyed hash algorithm on generation keys and seed, and an evaluation of the overall key setup time for proposed scheme.

First, two one-way hash algorithms, SHA-1 and MD5, are implemented. We take data stream of 64 bytes. As shown in Table 2, for SHA-1 the code consumes 128 bytes of RAM, 4048 bytes of ROM, and takes approximately 10.5 ms to produce a 160-bit hash of a 64-byte message. MD5 produces a 128-bit message digest for a given data stream. The code consumes 176 bytes of RAM, 12.5 KB of ROM, and takes approximately 5.75 ms to hash a message of 64 bytes using 64-byte blocks.

A keyed hash message authentication code (HMAC) is a MAC calculated using a cryptographic hash function in combination with a secret key. As with any MAC, it can be used to verify the data integrity and the authenticity of a message. However, in our proposed scheme, we use HMAC to generate key chains from generation keys. Any iterative cryptographic hash function, such as MD5 or SHA-1 may be used in the calculation of an HMAC; the resulting MAC algorithm is termed HMAC-MD5 or HMAC-SHA-1 accordingly, where size of the output is same as the underlying hash function. Further, both HMAC-SHA1 and HMAC-MD5 implementations were validated using the test cases given in [7].

We implement both HMAC-SHA1 and HMAC-MD5 algorithms. The HMAC-SHA1 code consumes 30 bytes of RAM, 4424 bytes of ROM, and takes approximately 21.9 ms to produce of MAC of 64 bytes of data stream, as shown in Table 2. Whereas the HMAC-MD5 code consumes 90 bytes of RAM, 12986 bytes of ROM, and takes approximately 12.2 ms to produce a MAC of 64 bytes of data.

The proposed scheme is implemented with both algorithms (HMAC-SHA1 and HMAC-MD5) for key generation. The memory consumption for HMAC-SHA1 (780 bytes RAM and 22.2 KB ROM) is less than HMAC-MD5 (840 bytes RAM and 30.7 KB ROM), as shown in Table 3. However, the time required to generate MAC using HMAC-SHA1 is greater than HMAC-MD5, as shown in Fig. 10. We generate keys from generation keys (8 bytes) and seed (28 Bytes) using both HMAC-MD5 and HMAC-SHA1. Figure 10 shows that as the number of keys are increased, the processing time for key-chain generation increases accordingly. However, the increase in HMAC-SHA1 is significantly greater than HMAC-MD5.

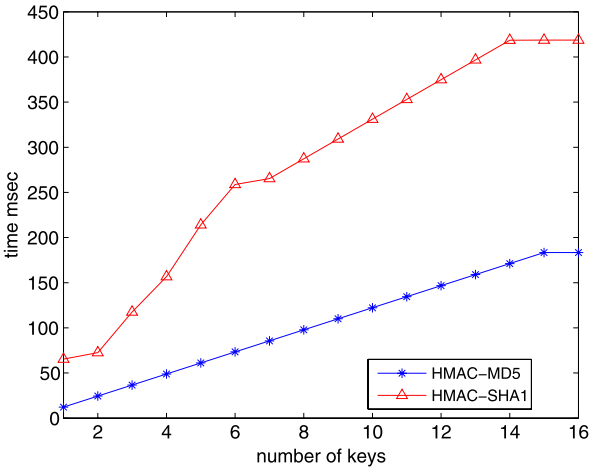

Fig. 10 The comparison of key generation process

Table 3 Memory usage for proposed scheme

Proposed scheme using	RAM (Bytes)	ROM (Bytes)
HMAC-SHA1	780 bytes	22206 bytes
HMAC-MD5	840 bytes	30728 bytes

One of the major constraints on implementing any scheme on a sensor platform is the small available payload size of packets. Specifically, under TinyOS, this limitation is commonly set to 29 bytes. As a result, all of the wireless interactions between nodes must adhere to this restriction. For instance, each of the generation keys deployed in L-sensors or H-sensors are of 8 bytes, which matches the key size used for the TinySec implementation of the RC5 block cipher [19]. We allocate 2 bytes for node identifier as well as 2 bytes for nonce value. We believe that 2 bytes (65,536 or 2^{16}) would be sufficient for a large sensor network. Further, for current battery lifetimes, 2 bytes for nonce would provide sufficient protection against nonce reuse. Authentication is provided by TinySec's CBCMAC and it occupies a total of 4 bytes. While not appropriate for other environments, an online attack of this authentication mechanism would require an average attack span of 20 months because of the limited bandwidth in this setting [19].

Figure 11 gives the details of the interfaces that are provided and used by the implementation of our proposed scheme in TinyOS. The component provides only one interface `Sensor`. The component uses either `SHA1` or `MD5` interface with `HMAC` interface. Further, we also use the interface `BlockCipher` provided by component `RC5` of TinySec. Moreover, there are several standard TinyOS interfaces used in the implementation, such as `Boot, Leds, Packet`, and `SplitControl`.

```
module SensorP {
  /* Provides Interfaces */
  provides interface Sensor;

  /* Uses Interfaces */
  /* Interfaces defined in Proposed Scheme */
  uses interface SHA1;
  uses interface MD5;
  uses interface HMAC;

  /* TinySec Interface */
  uses interface BlockCipher;

  /* Standard TinyOS interfaces */
  uses interface Boot;
  uses interface Leds;
  uses interface Packet;
  uses interface AMPacket;
  uses interface AMSend as RadioSend[am_id_t id];
  uses interface Receive as RadioReceive[am_id_t id];
  uses interface SplitControl as AMControl;
  uses interface Timer<TMilli> as Timer0;
  ...
}
```

Fig. 11 The *provides* and *uses* interfaces for the proposed scheme implementation in TinyOS

7 Conclusion

In this paper, we propose a key management scheme for heterogeneous sensor networks based on random key predistribution. In our scheme, instead of storing all the assigned keys in a sensor node, we store a small number of generation keys. Adversary or malicious nodes are precluded to join the cluster as each L-sensor is authenticated by CH using L-sensor's authentication key. In predeployment phase, each H-sensor is preloaded with the master key and L-sensor with authentication key (which is generated using master key). We also provide secure intercluster communication. Further, for scalable solution and easy maintenance, we provide dynamic nodes' addition and keys' revocation in case of node compromise. The results show that our scheme can significantly reduce the storage requirements as compared to other random key predistribution schemes. For instance, storage requirements can be reduced by 8 times as compared to AP [12], and 33 times as compared to basic scheme [14]. Also, the resiliency against node capture is better than previous key predistribution schemes. The TinyOS implementation shows that the proposed scheme can be efficiently implemented in real sensor networks. We compare both HMAC-SHA1 and HMAC-MD5 to generate key chains. The results show that although HMAC-SHA1 consumes less memory resources than HMAC-MD5, it is more computationally intensive.

Acknowledgements This work is in part supported by Higher Education Commission (HEC) Pakistan's International Research Support Initiative Program's scholarship given to Firdous Kausar to conduct her research at Acadia University, Canada. Further, we would like to thank National Science and Engineering Research Council (NSERC) Canada for their support in providing RTI and Discovery grants to Dr. Hussain at Acadia University, Canada.

References

1. Blom R (1982) Non-public key distribution. In: Advances in cryptology—CRYPTO 82, pp 231–236
2. Blundo C, De Santis A, Herzberg A, Kutten S, Vaccaro U, Yung M (1993) Perfectly-secure key distribution for dynamic conferences. In: CRYPTO '92: proceedings of the 12th annual international cryptology conference on advances in cryptology, London, UK, 1993. Springer, New York, pp 471–486. ISBN 3-540-57340-2
3. Bulusu V, Durresi A, Paruchuri V, Durresi M, Jain R (2006) Key distribution in mobile heterogeneous sensor networks. In: Global telecommunications conference, 2006. GLOBECOM '06. IEEE, New York, pp 1–5. ISBN 1-4244-0356-1
4. Chan H, Perrig A (2005) Pike: peer intermediaries for key establishment in sensor networks. In: INFOCOM 2005. 24th annual joint conference of the IEEE computer and communications societies, pp 524–535. ISBN 0-7803-8968-9
5. Chan H, Perrig A, Song D (2003) Random key pre-distribution schemes for sensor networks. In: IEEE symposium on security and privacy, May 2003, pp 197–213
6. Chan H, Perrig A, Song D (2003) Random key pre-distribution schemes for sensor networks. In: IEEE symposium on research in security and privacy, 2003
7. Cheng P, Glenn R (1997) Test cases for HMAC-MD5 and HMAC-SHA-1. In: RFC 2202, 1997
8. Cheng Y, Agrawal DP (2005) Efficient pairwise key establishment and management in static wireless sensor networks. In: Second IEEE international conference on mobile ad hoc and sensor systems, 2005
9. Di Pietro R, Mancini LV, Mei A (2003) Random key assignment secure wireless sensor networks. In: 1st ACM workshop on security of ad hoc and sensor networks, 2003
10. Du X, Lin F (2005) Maintaining differentiated coverage in heterogeneous sensor networks. EURASIP J Wirel Commun Netw (4):565–572
11. Du X, Xiao Y (2006) Energy efficient chessboard clustering and routing in heterogeneous sensor network. Int J Wirel Mobile Comput 1(2):121–130
12. Du X, Xiao Y, Guizani M, Chen H-H (2007) An effective key management scheme for heterogeneous sensor networks. Ad Hoc Netw 5(1):24–34
13. Duarte-Melo EJ, Liu M (2002) Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks. In: Proceedings of IEEE GLOBECOM, 2002
14. Eschenauer L, Gligor VD (2002) A key management scheme for distributed sensor networks. In: ACM CCS, 2002
15. Gay D, Levis P, von Behren R, Welsh M, Brewer E, Culler D (2003) The nesc language: a holistic approach to networked embedded systems. SIGPLAN Not 38(5):1–11. ISSN 0362-1340. doi:http://doi.acm.org/10.1145/780822.781133
16. Girod L, Stathopoulos T, Ramanathan N, Elson J, Estrin D, Osterweil E, Schoellhammer T (2004) A system for simulation, emulation, and deployment of heterogeneous sensor networks. In: 2nd international conference on embedded networked sensor systems, 2004
17. Goldreich O (2000) Foundations of cryptography: basic tools. Cambridge University Press, New York. ISBN 0521791723
18. Heinzelman WR, Chandrakasan A, Balakrishnan H (2000) Energy-efficient communication protocol for wireless microsensor networks. In: IEEE Hawaii int conf on system sciences, pp 4–7
19. Karlof C, Sastry N, Wagner D (2004) Tinysec: a link layer security architecture for wireless sensor networks. In: SenSys '04: proceedings of the 2nd international conference on embedded networked sensor systems. ACM Press, New York, pp 162–175. ISBN 1-58113-879-2. doi:http://doi.acm.org/10.1145/1031495.1031515
20. Lazos L, Poovendran R (2006) Stochastic coverage in heterogeneous sensor networks. ACM Trans Sensor Netw (TOSN) 2(3):325–358

21. Liu D, Ning P (2003) Location-based pairwise key establishments for static sensor networks. In: SASN '03: proceedings of the 1st ACM workshop on security of ad hoc and sensor networks. ACM, Press, New York, pp 72–82. ISBN 1-58113-783-4. doi:http://doi.acm.org/10.1145/986858.986869
22. Liu D, Ning P, Li R (2005) Establishing pairwise keys in distributed sensor networks. ACM Trans Inf Syst Secur 8(1):41–77. ISSN 1094-9224. doi:http://doi.acm.org/10.1145/1053283.1053287
23. Lu K, Qian Y, Hu J (2006) A framework for distributed key management schemes in heterogeneous wireless sensor networks. In: IEEE international performance computing and communications conference, pp 513–519
24. Ma Y, Dalal S, Alwan M, Aylor J (2003) Rop: a resource oriented protocol for heterogeneous sensor networks. In: Virginia tech symposium on wireless personal communications, 2003
25. Menezes AJ, van Oorschot PC, Vanstone SA (1996) Handbook of applied cryptography. CRC Press, New York. ISBN 08493-8523-7
26. Oliveira LB, Wong HC, Bern M, Dahab R, Loureiro AAF (2006) Sec leach: a random key distribution solution for securing clustered sensor networks. In: 5th IEEE international symposium on network computing and applications, pp 145–154
27. Perrig A, Szewczyk R, Tygar JD, Victorwen, Culler DE (2001) Spins: security protocols for sensor networks. In: Seventh annual int'l conf on mobile computing and networks, July 2001
28. Ren K, Zeng K, Lou W (2006) A new approach for random key predistribution in large-scale wireless sensor networks. Wirel Commun Mobile Comput 6(3):307–318
29. Traynor P, Kumar R, Bin Saad H, Cao G, La Porta T (2006) Establishing pair-wise keys in heterogeneous sensor networks. In: INFOCOM 2006. 25th IEEE international conference on computer communications. Proceedings, pp 1–12. ISBN 1-4244-0221-2
30. Traynor P, Kumar R, Bin Saad H, Cao G, La Porta T (2007) Efficient hybrid security mechanisms for heterogeneous sensor networks. IEEE Trans Mobile Comput 6(6):663–677. ISSN 1536-1233
31. Wadaa A, Olariu S, Wilson L, Eltoweissy M (2004) Scalable cryptographic key management in wireless sensor networks. In: Proceedings of the 24th international conference on distributed computing systems workshops—W7: EC (ICDCSW'04), Washington, DC, USA. IEEE Computer Society, Silver Spring, pp 796–802. ISBN 0-7695-2087-1
32. Woo A, Culler DE (2001) A transmission control scheme for media access in sensor networks. In: MobiCom '01: proceedings of the 7th annual international conference on mobile computing and networking. ACM Press, New York, pp 221–235. ISBN 1-58113-422-3. doi:http://doi.acm.org/10.1145/381677.381699
33. Zhu S, Xu S, Setia S, Jajodia S (2003) Establishing pairwise keys for secure communication in ad hoc networks: a probabilistic approach. In: 11th IEEE international conference on network protocols (ICNP'03)

Firdous Kausar is a Ph.D. candidate at the College of Signals, National University of Science and Technology, Pakistan. She has authored more than 7 papers including a book chapter. Her research interests are in network and data security, key management and authentication protocols, sensor networks, mobile and ad hoc networks. She is a member of IEEE and IACR (International Association of Cryptologic Research).

Sajid Hussain is an Assistant Professor in the Jodrey School of Computer Science, Acadia University, Canada. He received Ph.D. in Electrical Engineering from the University of Manitoba, Canada. Dr. Hussain is investigating sensor networks, communication protocols, security, database management, intelligent techniques and architectures for ubiquitous and pervasive applications such as smart homes and petroleum offshore facilities. He has published more than 35 refereed journal, conference and workshop papers. The research is financially supported by several grants and contracts, such as Natural Sciences and Engineering Research Council (NSERC) Canada, National Research Council (NRC) Canada, Atlantic Innovation Fund (AIF), and Nova Scotia Health Research Foundation (NSHRF).
Dr. Hussain has co-organized several International workshops and conferences, served on many technical program committees, and reviewed papers for several journals, conferences and workshops. Further, he has reviewed grant proposals for NSERCs Discovery Grants, Strategic Project Grants (SPG), and Research Tools and Instrument (RTI) Grants. He is a member of IEEE and ACM societies.

Laurence T. Yang research includes high performance computing and networking, embedded systems, ubiquitous/pervasive computing and intelligence. He has published around 250 papers in refereed journals, conference proceedings and book chapters in these areas. He has been involved in more than 100 conferences and workshops as a program/general conference chair and more than 200 conference and

workshops as a program committee member. He served as the vice-chair of IEEE Technical Committee of Supercomputing Applications (TCSA) until 2004, currently is in the executive committee of IEEE Technical Committee of Scalable Computing (TCSC), and of IEEE Technical Committee of Self-Organization and Cybernetics for Informatics, and of IFIP Working Group 10.2 on Embedded Systems, and of IEEE Technical Committee of Granular Computing. He is also the co-chair of IEEE Task force on Intelligent Ubiquitous Computing.
In addition, he is the editors-in-chief of 9 international journals and few book series. He is serving as an editor for around 20 international journals. He has been acting as an author/co-author or an editor/co-editor of 30 books from Kluwer, Springer, Nova Science, American Scientific Publishers and John Wiley & Sons. He has received three Best Paper Awards including the IEEE 20th International Conference on Advanced Information Networking and Applications (AINA-06); one IEEE Best Paper Award, 2007; one IEEE Outstanding Paper Award, 2007; Distinguished Achievement Award, 2005; Distinguished Contribution Award, 2004; Outstanding Achievement Award, 2002; Canada Foundation for Innovation Award, 2003; University Research/Publication/Teaching Award 00-02/02-04/04-06.

Ashraf Masood is a Head of Research and Development Establishment, College of Signals, National University of Science and Technology, Pakistan. His research interests are in cryptography, network security, sensor networks, cryptanalysis, stream ciphers, algebraic attack.