Network Decoupling for Secure Communications in Wireless Sensor Networks

Wenjun Gu, Xiaole Bai, Sriram Chellappan and Dong Xuan
Department of Computer Science and Engineering
The Ohio-State University, Columbus, Ohio 43210–1277
Email: {gu, baixia, chellapp, xuan}@cse.ohio-state.edu

Abstract— **Secure communications are highly demanded by many wireless sensor network (WSN) applications. The random key pre-distribution (RKP) scheme has become well accepted to achieve secure communications in WSNs. However, due to its randomness in key distribution and strong constraint in key path construction, the RKP scheme can only be applied in highly dense networks, which are not always feasible in practice. In this paper, we propose a methodology called *network decoupling* to solve this problem. With this methodology, a wireless sensor network is decoupled into a *logical key-sharing network* and a *physical neighborhood network*, which significantly releases the constraint in key path construction of RKP scheme. We design a secure neighbor establishment protocol (called RKP-DE) as well as a set of link and path dependency elimination rules in decoupled wireless sensor networks. Our analytical and simulation data demonstrate the performance enhancement of our solution and its applicability in non-highly dense wireless sensor networks.**

I. Introduction

Wireless Sensor Networks (WSNs) are gaining wide acceptance today. A host of new applications are being realized that involve many tiny wireless sensors performing sensing and communication tasks. Many of these applications are in hostile environments, and their success is contingent on preventing the WSNs information from being accessible to external malicious attackers. In this paper, we address the issue of providing secure communications in WSNs.

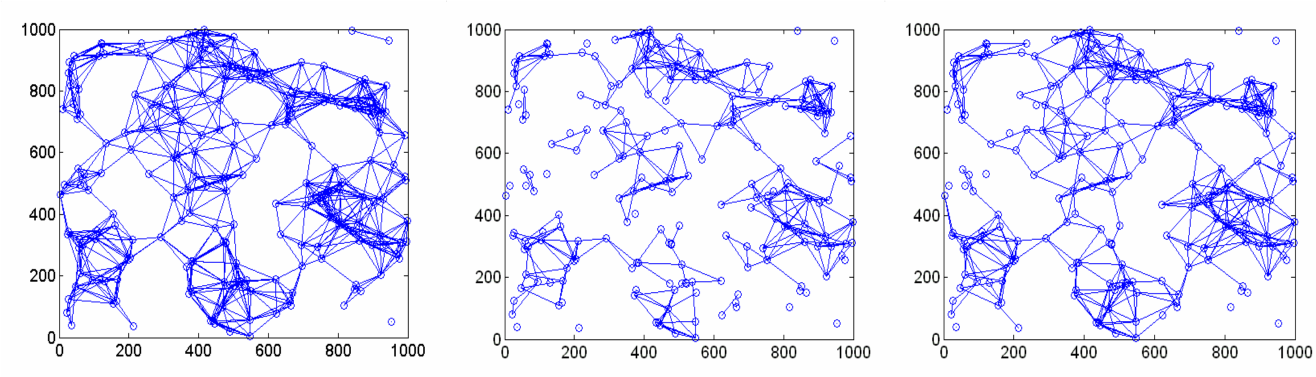
Motivation: A host of key distribution techniques have been proposed to achieve secure communications in traditional wired networks and wireless ad hoc networks. However, they cannot be applied in WSNs due to the unique characteristics of WSNs like network scale, ease of node capture, physical constraints in energy and memory, etc. For instance, the traditional public key cryptography [1], [2] is too energy consuming to be carried out by energy constrained sensors. The key distribution center based scheme [3] is centralized and not scalable when network size increases. Other techniques like using a single master key for all communication or establishing unique pair-wise keys between each pair of nodes are either too vulnerable under attack or may require too much memory, which are all unsuitable in WSNs.

In order to address the above concerns, the seminal scheme based on *Random Key Pre-distribution* (RKP in short) was first proposed in [4]. Each sensor is initially pre-distributed with a small number of k distinct keys randomly chosen from a larger key pool of K keys. Two nodes within communication range of each other (called *physical neighbors*) can directly establish a pair-wise key between them if they share at least one pre-distributed key. Alternatively, two sensors can establish a pair-wise key indirectly through a key path traversing through other sensors, with the constraint that any two physically neighboring sensors on this path share at least one pre-distributed key. For the rest of the paper, physical neighbors that have established a pair-wire key are called *secure neighbors*. The RKP scheme is widely accepted in WSNs due to its simplicity, low overhead, scalability and energy efficiency. As such, it has served as a foundation for a host of key management protocols in WSNs that aim towards improving the probability of pair-wise key establishment, enhancing the resilience to node capture, or decreasing storage overhead [5], [6], [7], [8], [9], etc.

However, all the RKP based schemes have an inherent limitation. The performance of RKP is satisfactory only in highly dense sensor networks, where the average number of physical neighbors per node (i.e., average physical node degree) $>= 20$ [4], [5], [6]. As we know, such a high density is not always feasible in practice. In fact, due to the randomness in key distribution and strong constraint in key path construction, it often happens that many physical neighbors cannot become secure neighbors, i.e. the secure node degree is very low, in non-highly dense networks. Consequently, they will have low secure connectivity and are very likely to be partitioned. Fig. 1 illustrates this. The original network is shown in Fig. 1 (a). There is an edge between two nodes if they are physical neighbors. The average physical node degree is set as 9.71. The corresponding secure network generated by RKP is shown in Fig. 1 (b) where an edge exists between two nodes if they are secure neighbors. The average secure node degree in Fig. 1 (b) is only 4.06. It is much smaller compared to the average physical node degree. As can be seen, the network in Fig. 1 (b) is partitioned into many connected components. Two nodes cannot communicate securely if they reside in different connected components. The RKP based schemes when applied to non-highly dense networks have poor performance.

Our Contributions: In this paper, we aim to solve the above problem. Our contributions are three-fold.

- *Network Decoupling:* We propose a methodology called *network decoupling* for secure communications in wire-



(a) Original Network

(b) Network with *RKP*

(c) Network with *RKP-DE*

Fig. 1. Average secure node degree comparison between RKP and RKP-DE when at most one proxy is used on each key path. Our RKP-DE achieves 40% improvement in average secure node degree. The network is of size $1000m$ * $1000m$, where 200 nodes are deployed uniformly at random. All nodes have a communication range of $133m$ and the average physical node degree is 9.71. We set $K = 10000$ and $k = 50$.

less sensor networks. In random key pre-distributed sensor networks, there exist two types of relationships between any two nodes. One is logical (sharing pre-distributed keys), and the other is physical (within communication range). In network decoupling, we decouple these two relationships in the sensor networks. As such, for any two nodes connected logically, we can independently find a path for them physically and vice versa. The flexibility offered by decoupling greatly enables finding more logical and physical paths, thereby enhancing the chances of pair-wise key establishment between physical neighbors in the network.

- *Protocol Design:* Based on network decoupling methodology, we design a new protocol for secure neighbor establishment between physical neighbors in the decoupled network. We call our protocol as the RKP-DE protocol, where logical key paths are constructed based on key sharing information. Then, corresponding physical key paths are constructed based on node neighborhood information in our protocol.
- *Dependency Elimination:* Our third contribution is proposing novel dependency elimination rules in our protocol to detect and eliminate key dependencies at link and path level without compromising existing resilience. In key establishment, when multiple key paths are constructed, there is a possibility of some links (or paths) being dependent on other links (or paths). Such dependencies introduce unnecessary overhead in terms of communication and computation. We point out that such dependencies exist in all existing RKP based protocols, where multiple key paths are used [5], [6], [9]. Our dependency elimination rules can be applied to them to minimize their overhead as well.

To illustrate performance improvement of our RKP-DE protocol, in Fig. 1 (c), we show the secure network generated by our RKP-DE protocol. The average secure node degree in Fig. 1 (c) has now increased to 5.68, a 40% improvement over that in Fig. 1 (b). As our analysis shows in Section IV, the average secure node degree improvement of RKP-DE over RKP is around 40% when one proxy is used on each key path. With increase in average secure node degree, the quality of secure communications naturally increases, demonstrating the benefits of network decoupling, as also shown in our performance evaluations in Section V.

We wish to point out that the methodology of decoupling in itself is not new in networking. In [10], connection establishment is decoupled from QoS reservation to enhance the efficiency of frequent short lived Internet connections. The benefits of decoupling policy from mechanisms in Internet routing have been demonstrated in [11]. In [12], an approach is proposed that decouples control from data in TCP congestion control. Another work is [13], where path naming is decoupled from the actual path to enable better data delivery in dense sensor networks. However, to the best of our knowledge, our work is the first one that applies this methodology for secure communications in wireless sensor networks.

The remaining of our paper is organized as follows. We discuss random key pre-distribution and other related works in Section II. The methodology of network decoupling is introduced in Section III, and our secure neighbor establishment protocol is detailed in Section IV. In Section V, we present performance evaluations, and finally we conclude our paper in Section VI.

II. The Random Key Pre-distribution Protocol in Wireless Sensor Networks

In a seminal work in [4], the idea of random key pre-distribution (RKP) was first proposed to establish pair-wise keys in WSNs. Before nodes are deployed randomly in the network, each node is pre-distributed with k distinct keys randomly chosen from a large key pool of size K. The set of keys pre-distributed in node a is called the *key chain* of node a, denoted by $KC(a)$. In Fig. 2, nodes b, c, d and e are four physical neighbors of node a. Each node is pre-distributed with three keys, which are listed beside the corresponding node. A solid line exists between two nodes if they are physical

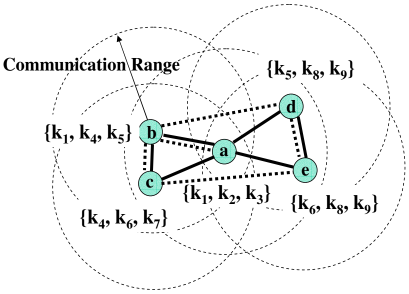


Fig. 2. Pair-wise key establishment in RKP protocol.

neighbors (within communication range), and a dashed line exists between two nodes if they are logical neighbors (share at least one pre-distributed key).

After deployment, each node sends a message to its physical neighbors, containing its node ID and the key IDs of its pre-distributed keys. If node a shares at least one pre-distributed key with a physical neighbor, a pair-wise key between them can be established directly, such as nodes a and b in Fig. 2. To do so, node a can send a randomly generated pair-wise key to node b with the pair-wise key encrypted using their shared key k_1. If node a does not share any key with a physical neighbor, such as node c, node a will attempt to establish a pair-wise key indirectly using other nodes as *proxies*. Here, a key path is attempted to be constructed comprising of one or multiple proxies, where any two successive nodes on the key path are physical neighbors and share at least one pre-distributed key. The pair-wise key generated by node a is sent to its physical neighbor on the key path, with the constraint that the pair-wise key is encrypted/decrypted in *each* hop till it reaches the destination. That is, the logical (sharing pre-distributed keys) constraint and the physical (within communication range) constraint are coupled together during key path construction. In Fig. 2, node b can be the proxy between nodes a and c. The pair-wise key between nodes a and c is first generated by node a, then it is sent to node b encrypted by key k_1. Node b will decrypt the pair-wise key, encrypt it by key k_4 and send to node c. Finally node c decrypts the pair-wise key, and uses it to encrypt future direct communication with node a.

The standard attack model used in analyzing secure communications is one where the attacker does not attempt to disrupt network operation; rather it attempts to decipher as much information as possible from sensor communications [4], [5], [7]. As such, the attacker will typically launch two types of attacks: *link monitor attack* and *node capture attack*. In the link monitor attack, the attacker monitors and records all the wireless communications in the network *immediately* after node deployment. In the node capture attack, the attacker will physically capture a certain number of sensors after node deployment. Once a node is captured, its pre-distributed keys are disclosed to the attacker. Combining the pre-distributed keys disclosed and the messages recorded, the attacker will be able to infer the pair-wise keys between some neighboring nodes, even if the nodes themselves are not captured. The pair-wise keys inferred by the attacker are $compromised$, as is the corresponding secure communications between those neighboring nodes.

To evaluate the performance of RKP protocol, two types of metrics are considered. The first is *connectivity*, which includes *local connectivity* and *global connectivity*. Local connectivity is defined as the probability that two physically neighboring nodes are able to establish a pair-wise key in between. Global connectivity is defined as either the probability that the whole secure network (an example is shown in Fig. 1 (b) or (c)) is connected, or the percent of nodes in the largest connected component of the secure network. The other performance metric is *resilience*, which is defined as the probability that a pair-wise key between two nodes is not compromised given that those two nodes are not captured. The overall goal clearly is to make connectivity and resilience as high as possible.

The RKP protocol [4] has received wide acceptance in WSNs due to its simplicity, low overhead, scalability and energy efficiency. It has served as a foundation for many other works based on random key pre-distribution, aiming to improve performance or lower overhead [5], [6], [7], [8], etc. In [5], the performance of the basic RKP protocol is enhanced by constructing multiple key paths using proxies for pair-wise key establishment between physically neighboring nodes. With multiple key paths, as long as at least one key path is uncompromised, the pair-wise key is secure. Similarly, [6] uses multiple two hop key paths to enhance resilience further under a slightly weaker attack model. We point out that in both works, a very high network density (average physical node degree between 20 and 250) is assumed to achieve satisfactory performance.

Several other works orthogonally improve the basic RKP protocol by extending the key structure, exploiting certain network properties to enhance performance, or decreasing overhead. In [7] and [8], the authors independently extend the basic RKP protocol by pre-distributing key structures (either polynomials or vectors) instead of keys to establish pair-wise keys. When the number of captured nodes is small, this protocol has much better resilience compared to the basic protocol. Other works like [14], [15], [16] use power control, channel diversity or network hierarchy to enhance performance under assumptions on sensor hardware, network topology etc. Recently, some works have used deployment knowledge to achieve comparable performance with fewer number of keys pre-distributed [17], [18], [19]. These works rely on the assumption that positions of neighboring nodes in the network are partially known a priori, helping in decreasing the number of keys pre-distributed to achieve comparable performance. We point out that our methodology of network decoupling is orthogonal to all the works above, and can complement them to achieve further performance improvement and overhead reduction.

III. Network Decoupling in Random Key Pre-distributed Sensor Networks

A. Network Decoupling

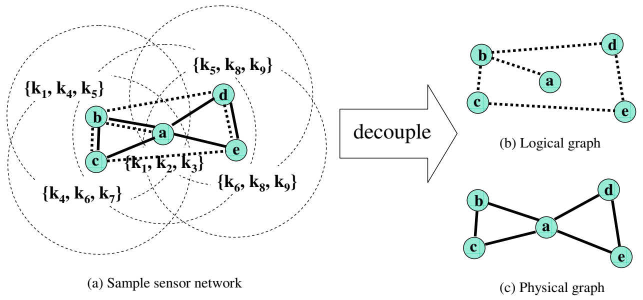
In random key pre-distributed sensor networks, there exist two types of relationships between any two nodes. One is logical (sharing pre-distributed keys), and the other is physical (within communication range). We can separate these two types of relationships by decoupling a random key pre-distributed sensor network into two graphs: a logical one and a physical one. Two nodes in the logical graph have an edge between them if they share at least one pre-distributed key. Similarly two nodes in the physical graph have an edge between them if they are within communication range of each other. In the example of Fig. 3 (a), node a shares a key with node b. Consequently, nodes a and b will have an edge between them in the logical graph. Besides, node a is within the communication range of the other four nodes. Consequently in the physical graph, there will be an edge from node a to the other four nodes. For the example in Fig. 3 (a), its decoupled logical and physical graphs are shown in Fig. 3 (b) and (c) respectively. Detailed description on how nodes construct these graphs is presented in Section IV.

In random key pre-distributed sensor networks, we define secure communication as the communication between two nodes where all messages transmitted (possibly via multi-hops) are encrypted. Now we will show how network decoupling helps achieve secure communication. There are two cases possible, where two nodes in the network can communicate securely. The first case is where the two communicating nodes share a pre-distributed key (i.e., they are *directly* connected in the logical graph) and the nodes are also connected (via one or multiple hops) in the physical graph. In this case, the source node can encrypt the messages using the shared pre-distributed keys, and each intermediate node in the physical graph can simply forward the messages towards the destination, which will decrypt the messages using the shared pre-distributed keys. The second case is one where the two communicating nodes do not share a pre-distributed key (i.e., they are *not directly* connected in the logical graph), but are connected *indirectly* in the logical graph, and the two nodes for each logical hop are connected (directly or indirectly) in the physical graph. In this case, encryption occurs at each intermediate node in the logical graph, while each intermediate node in the physical graph simply forwards the messages.

We point out that in order to apply decoupling, each sensor needs to know both the key sharing and node neighborhood information among its physical neighbors. Note that it will incur significant communication overhead to obtain such information on a global scale. Hence our network decoupling is a purely localized behavior, where each node obtains local information and constructs its local logical and physical graphs in a distributed way.

B. Analysis

In this section, we will demonstrate the benefit of network decoupling quantitatively by analysis. Specifically, we will derive the probability for the case where two physically neighboring nodes are able to communicate securely. As a matter of fact, this probability is also the probability that two physically neighboring nodes are able to establish a pair-wise key via secure communication. Due to space limitations, we only present the analysis for the case where at most one proxy is used on a key path. Interested readers are referred to [20] for the analysis in general case.



(a) Sample sensor network

(b) Logical graph

(c) Physical graph

Fig. 3. Decouple a sensor network into a logical graph and a physical graph.

For two physically neighboring nodes to communicate securely, there exist three possible situations: *(1)* The two nodes share pre-distributed keys (directly connected in the logical graph), such as nodes a and b in Fig. 3 (a). Clearly, they can achieve secure communication directly. We denote the probability that this situation happens as P_1. *(2)* The two nodes do not share pre-distributed key (not directly connected in the logical graph), but they have a common physical neighbor that shares pre-distributed keys with both of them. In Fig. 3 (a), nodes a and c do not share pre-distributed key, but have a common physical neighbor node b that shares one pre-distributed key with both of them. Secure communication between nodes a and c can now be achieved via the help of node b acting as a proxy. We denote the probability that this situation happens as P_2. *(3)* The two nodes do not share pre-distributed key (not directly connected in the logical graph), and they cannot find a proxy satisfying the second situation above. But there exists a proxy that shares pre-distributed keys with both of those two nodes, and is a physical neighbor of only one of them. In Fig. 3 (a), nodes a and d do not share pre-distributed key, but node b shares one pre-distributed key with both of them, and node b is a physical neighbor of only node a. Secure communication between nodes a and d can be achieved via the help of node b acting as a proxy. We denote the probability that this situation happens as P_3.

Let us define *coupled network* as the network in which the logical constraint and the physical constraint are always satisfied simultaneously for each hop on a secure communication path. Therefore two nodes in a coupled network can achieve secure communication if and only if either of the first two situations happens. In the third situation, secure communication is not possible in a coupled network. On the other hand, in a decoupled network, secure communication is possible if any of the three situations happens. We denote the probability that two nodes can achieve secure communication using at most one proxy on each key path in coupled and decoupled network by P_{couple} and $P_{decouple}$ respectively. Since the above three situations are disjoint, the expressions of P_{couple} and $P_{decouple}$ are simply given by,

$$P_{couple} = P_1 + P_2, \tag{1}$$

$$P_{decouple} = P_1 + P_2 + P_3. \tag{2}$$

Clearly, $P_{decouple} > P_{couple}$. This demonstrates that network decoupling enhances the chance for two neighboring nodes to communicate securely. In the following, we will derive the expressions for P_1, P_2 and P_3.

Recall that P_1 is the probability that two nodes share at least one pre-distributed key. It is given by,

$$P_1 = 1 - \binom{K}{2k} \cdot \binom{2k}{k} / \binom{K}{k}^2. \tag{3}$$

If D_p denotes the average physical node degree, the average number of nodes in the overlapped communication ranges of two physically neighboring nodes is $0.5865D_p$ [5]. The probability that n_1 nodes in the overlapped communication ranges of both nodes share pre-distributed keys with one of those two nodes is $\binom{0.5865D_p}{n_1}(P_1)^{n_1}(1-P_1)^{0.5865D_p - n_1}$. The probability that at least one of the above n_1 nodes shares pre-distributed keys with the other node is $1-(1-P_1)^{n_1}$. Therefore, P_2 is given by,

$$P_2 = (1-P_1) \cdot \sum_{n_1=1}^{0.5865D_p} \Big(\binom{0.5865D_p}{n_1}(P_1)^{n_1} (1-P_1)^{0.5865D_p - n_1} \cdot (1-(1-P_1)^{n_1})\Big). \tag{4}$$

For two physically neighboring nodes, the average number of nodes in the communication range of one node but outside the communication range of the other node is $2(D_p - 0.5865D_p) = 0.8270D_p$. Similarly, P_3 is given by,

$$P_3 = (1-P_1) \cdot (1-P_2) \cdot \sum_{n_1=1}^{0.8270D_p} \Big(\binom{0.8270D_p}{n_1} (P_1)^{n_1}(1-P_1)^{0.8270D_p - n_1} \cdot (1-(1-P_1)^{n_1})\Big). \tag{5}$$

We point out that P_{couple} and $P_{decouple}$ (in (1) and (2)) are also the probabilities that two physically neighboring nodes are able to establish a pair-wise key between them (via one proxy at most) in a coupled and decoupled network respectively. The derivations of them will be used later in the analysis in Section IV.

IV. Secure Neighbor Establishment Protocol in Decoupled Networks

A. Overview

In this section, we discuss the design of our new protocol for establishing secure neighbors in decoupled random key pre-distributed sensor networks. We call our protocol as RKP-DE protocol. The protocol has four major components in its execution: 1) constructing local logical and physical graphs in the decoupled network for each node, 2) establishing multiple physical key paths between physically neighboring nodes, 3) eliminating dependencies among the multiple key paths, and 4) establishing pair-wise keys between physically neighboring nodes. The RKP-DE protocol is distributed in its execution like the traditional RKP protocol. Similar to the model in the traditional scheme in [4], the network model we consider is one where a set of n sensors are deployed randomly. Each sensor is pre-distributed with k distinct keys randomly chosen from a key pool of size K.

The major differences between our RKP-DE protocol and the traditional RKP protocol are due to the first three components. In the traditional RKP protocol, key paths are established in a network where the physical and logical graphs are coupled. On the other hand in our RKP-DE protocol, the physical and logical graphs are separated/decoupled. The first component of our RKP-DE protocol is each node constructing these two local graphs decoupled from each other. The local logical graph is constructed based on key sharing information and the local physical graph is constructed based on node neighborhood information, following the methodology of network decoupling discussed earlier in Section III. The second component in our RKP-DE protocol is to establish logical key paths between two physically neighboring nodes based on the logical graph, and for these logical key paths, corresponding physical key paths are established based on the physical graph. The decoupling feature enables more key paths (both logical and physical) to be constructed when compared to the traditional RKP protocol. Note that when multiple key paths (each with multiple links/hops) are constructed, there is a possibility of some links (or paths) being dependent on other links (or paths). Such dependencies introduce unnecessary overhead in terms of communication and computation. The third component in our RKP-DE protocol proposes novel dependency elimination rules to detect and eliminate such dependencies without compromising the existing resilience. Each component in our RKP-DE protocol is described in detail below.

B. Local Graphs Construction

After node deployment, each node obtains the key sharing and node neighborhood information within its communication range by local communication with its physical neighbors. We assume that from local communication, each node can determine whether any two of its physical neighbors are physical neighbors or not. This can be easily done by exchanging neighbor information during initial communication. With this information, each node constructs a local logical graph (G_l) and a local physical graph (G_p). In the local logical graph, two nodes are connected if they share at least one pre-distributed key, while in the local physical graph, two nodes are connected if they are within communication range of each other. Note that our protocol needs only local information exchange and is purely distributed. In this paper, we assume each node obtains the local information within its communication range (one-hop). Information across multiple hops can be obtained by further information exchange, but will incur more communication overhead.

C. Key Paths Construction

Algorithm 1 shows the pseudocode of key paths construction executed by each node in the network. In Algorithm 1, u denotes an arbitrary node, while $G_l(u)$ and $G_p(u)$ are its local logical and physical graphs respectively. Initially the logical key path tree of node u (T_u) is empty. The key paths construction is executed in two steps as shown in Algorithm 1. First, T_u is constructed by node u based on its local logical graph $G_l(u)$ (lines 1 to 7). This logical key path tree T_u contains all the logical key paths between u and all its secure neighbors. Then, node u constructs corresponding physical key paths based on both T_u and its local physical graph $G_p(u)$ (lines 8 to 13). The dependency checking in line 3 and 11 will be discussed in the next subsection.

Logical key path tree construction: The protocol constructs logical key path tree (lines 1 to 7) using a variant of the standard depth-first-search algorithm, in which a node could be chosen multiple times (on different paths). Here $N(u)$ denotes the set of physical neighbors of node u. Fig. 4 shows the resultant logical key path tree for node a in the example of Fig. 3 (b). By executing the algorithm just once on its local logical graph in Fig. 3 (b), node a is able to obtain all logical key paths to all its neighbors. Taking node e as an example, node a obtains two logical key paths between node a and node e, that are $< a, b, c, e >$ and $< a, b, d, e >$.

Physical key paths construction: After obtaining the logical key path tree (T_u), node u begins to construct physical key paths for its neighbors (lines 8 to 13). For each physical neighbor v, node u first obtains a set of logical key paths between u and v (T_{uv}) from T_u. Out of all such key paths in T_{uv}, some of them will be eliminated based on dependency checking (as discussed in the next subsection). The set of paths that pass the dependency checking is denoted as T'_{uv}. Finally, for all logical key paths in T'_{uv}, corresponding physical key paths (T^*_{uv}) are obtained. In Fig. 3 (b), the logical key path $< a, b, d, e >$ contains a logical hop $< b, d >$ between two non-neighboring nodes. From Fig. 3 (c), we see that a physical path $< b, a, d >$ can replace the above logical hop. Therefore, for logical key path $< a, b, d, e >$, its corresponding physical key path is $< a, b, a, d, e >$, in which each hop is between two physically neighboring nodes. Message encryption/decryption occurs for each logical hop, while message transmission occurs for each physical hop. Here, we select the physical path with fewest hops to replace a logical hop between non-neighboring nodes. Other policies can be chosen if energy consumption, load balancing, etc. are to be considered.

D. Dependency Elimination

We now discuss elimination of link and path dependencies in steps 3 and 11 of Algorithm 1. Generally, if more key paths are used, resilience is enhanced. This is because when multiple key paths exist between two nodes, the attacker needs to compromise all key paths in order to compromise the pair-wise key established between them. However, this is not always true. Existing links (or paths) may have dependencies among them such that the compromise of some links (or paths) automatically leads to the compromise of other dependent links (or paths). Clearly, the presence of such dependency does not enhance resilience. They only increase overhead in terms of both storage and energy consumption (due to communication and computation). In this subsection, we propose two novel *dependency elimination rules* to decrease such overheads without affecting the resilience of the pair-wise keys established.

Algorithm 1 Pseudocode of Key Paths Construction

```
1:  Log_Key_Path_Tree_Construct($u$,$G_l(u)$,$T_u$)
2:  for each $v \in N(u)$
3:      if $Link\_Dependency\_Checking(v, u, T_u) == PASS$, then
4:          $Insert(u, v, T_u)$;
5:              $Log\_Key\_Path\_Tree\_Construct(v, G_l(u), T_u)$;
6:      end if
7:  end for

8:  Phy_Key_Paths_Construct($u$,$G_p(u)$,$T_u$)
9:  for each $v \in N(u)$
10:     $obtain\ the\ set\ of\ all\ logical\ key\ paths\ between\ u\ and\ v\ (T_{uv})\ from\ T_u$;
11:     $T'_{uv} = Path\_Dependency\_Checking(T_{uv})$;
12:     $obtain\ the\ corresponding\ set\ of\ physical\ key\ paths\ T^*_{uv}\ from\ T'_{uv}$;
13: end for

14: Insert($u$,$v$,$T_u$)
15:     $Insert\ node\ v\ into\ T_u\ as\ a\ child\ of\ node\ u.$
```

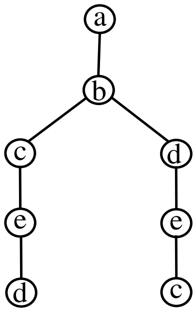


Fig. 4. Logical Key Path Tree of Node a

1) Link Dependency Elimination: We illustrate link dependency with an example in Fig. 5. Node a obtains a logical key path $< a, \cdots, c, d, \cdots, e, f, \cdots, b >$ to its physical neighbor node b. If we denote $K(i, j)$ as the set of shared pre-distributed keys used to encrypt the messages on the logical hop $< i, j >$ in a logical key path, there exists a link dependency between the hops $< c, d >$ and $< e, f >$ in that $K(c, d) \subseteq K(e, f)$. Since both nodes c and f share keys k_1 and k_2, there must exist another shorter logical key path $< a, \cdots, c, f, \cdots, b >$, which

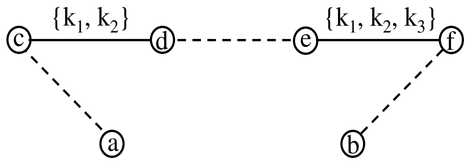


Fig. 5. Link Dependency Example

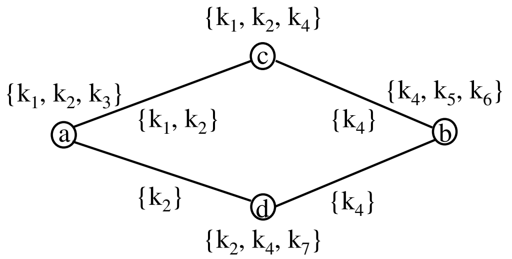


Fig. 6. Path Dependency Example

has better resilience than the original one. This is because the compromise of any logical hop between nodes d and e will compromise the original key path definitely, while it is possible that the shorter key path is not compromised. On the other hand, the compromise of the shorter key path will definitely compromise the original key path[1]. Also, using a shorter key path will save overhead. We formally define link dependency as follows.

Link Dependency: Given two logical hops $< i_1, j_1 >$ and $< i_2, j_2 >$ in a logical key path, there exists link dependency between these two hops if either $K(i_1, j_1) \subseteq K(i_2, j_2)$ or $K(i_2, j_2) \subseteq K(i_1, j_1)$.

Our *link dependency elimination rule* is that once such a link dependency is detected on a logical key path, the protocol will eliminate that logical key path. In the above example, the logical key path $< a, \cdots, c, d, \cdots, e, f, \cdots, b >$ will be eliminated since a shorter key path $< a, \cdots, c, f, \cdots, b >$ with better resilience exists. The pseudocode of link dependency checking is given in Algorithm 2 (lines 1 to 7). In Algorithm 2, *root* denotes the root node of the logical key path tree T, $Path(u, root)$ denotes the set of nodes on the logical key path from u to *root*, and *v.parent* denotes the parent node of node v on the tree T. As we can see in Algorithm 2, link dependency will be checked to output a $PASS$ or $FAIL$, which is returned in line 3 of Algorithm 1.

2) Path Dependency Elimination: Apart from link dependency, another type of dependency called path dependency may exist. In Fig. 6, there are two logical key paths between nodes a and b. However, we can see that the compromise of the key path $< a, c, b >$ (disclosure of keys (k_1, k_2) or (k_4)) always leads to the compromise of the other key path $< a, d, b >$, but not vice versa. Therefore, given that key path $< a, c, b >$ exists, the other key path $< a, d, b >$ becomes redundant in terms of resilience, and also incurs overhead. Denoting the set of logical hops on a logical key path p as L_p, and denote the set of shared pre-distributed keys used on a logical hop h as $K(h)$, path dependency is formally defined as follows.

Algorithm 2 Pseudocode of Dependency Checking

```
1:  Link_Dependency_Checking($v$,$u$,$T$)
2:  if $\exists$ node $w \in$ Path($u$,$root$), s.t. $K(v, v.parent) \subseteq K(w, w.parent)$, then
3:      $return\ FAIL$;
4:  else if $\exists$ node $w \in$ Path($u$,$root$), s.t. $K(w, w.parent) \subseteq K(v, v.parent)$, then
5:      $return\ FAIL$;
6:  else $return\ PASS$;
7:  end if

8:  Path_Dependency_Checking($T_{uv}$)
9:  $T'_{uv} = T_{uv}$;
10: while $\exists$ paths $p$ and $q \in T'_{uv}$, s.t. $p$ is weaker than $q$ OR $q$ is weaker than $p$, do
11:     if $p$ is weaker than $q$, then
12:         $T'_{uv} = T'_{uv} \setminus p$;
13:     else if $q$ is weaker than $p$, then
14:         $T'_{uv} = T'_{uv} \setminus q$;
15:     end if
16: end while
17: $return\ T'_{uv}$;
```

Path Dependency: Given two logical key paths p and q, there exists path dependency between p and q if either of the following two conditions is satisfied. *(1)* $\forall$ *logical hop* $h \in L_q$, $\exists$ *a logical hop* h' $(h' \in L_p)$, *s.t.* $K(h') \subseteq K(h)$; *(2)* $\forall$ *logical hop* $h \in L_p$, $\exists$ *a logical hop* h' $(h' \in L_q)$, *s.t.* $K(h') \subseteq K(h)$.

If the first condition of path dependency is satisfied, we call path p *weaker* than path q. Similarly, path q is *weaker* than path p if the second condition is satisfied. Our *path dependency elimination rule* is that after detecting path dependency between two logical key paths, our protocol will eliminate the weaker one. In the above example, the logical key path $< a, d, b >$ will be eliminated. In case two paths satisfy both conditions in the path dependency, we can eliminate one of them based on certain policies (e.g., the path with more physical hops). The pseudocode of path dependency checking is given in Algorithm 2 (lines 8 to 17).

E. Pair-wise Key Establishment

Once the physical key paths are constructed after dependency elimination, each sensor will generate distinct key shares at random, and send each key share on each physical key path for each secure neighbor. The messages are transmitted at each physical hop, while they are decrypted/encrypted at each logical hop. Take the logical key path $< a, b, d >$ in Fig. 3 (b) as an example. Its corresponding physical key path is $< a, b, a, d >$. We assume a key share $k_a^{(1)}$ is transmitted on this key path. We denote $\{M\}_k$ as the

[1] We formally prove that the shorter key path improves resilience over the original one in [20].

message M encrypted with key k. The key share transmission is executed as follows:

$$a \mapsto b : \{< a >, < a, d >, < k_a^{(1)} >\}_{k_1},$$
$$b \mapsto a : \ \ < d >, \{< a >, < d >, < k_a^{(1)} >\}_{k_5},$$
$$a \mapsto d : \{< a >, < d >, < k_a^{(1)} >\}_{k_5}.$$

In the message that node a sends to node b, $< a >$ denotes the source node, and $< a, d >$ denotes the remaining physical key path. In the message that b sends to a, the $< d >$ without encryption denotes the remaining physical key path since a cannot decrypt the message using key k_5.

Similarly, node a can transmit another random key share $k_a^{(2)}$ on another logical key path $< a, b, c, e, d >$ to node d. Node d may also construct other key paths [2], and transmit its key shares to node a. Finally, nodes a and d can compute a common pair-wise key via some simple operation such as bit-wise XOR operation, based on all the key shares they both generated.

F. Analysis

In this section, we will derive the expression for the average secure node degree in both RKP protocol and RKP-DE protocol, denoted by D_s^{RKP} and D_s^{RKP-DE} respectively. Due to space limitations, we only present the derivation for the case where one proxy is used on each key path. Interested readers are referred to [20] for the general case analysis where arbitrary number of proxies are used on each key path. In Section III, we derived expressions for P_{couple} and $P_{decouple}$ in (1) and (2) respectively, which denote the probability that two physically neighboring nodes are able to construct a key path in RKP-DE protocol and RKP protocol with at most one proxy respectively. With this, we can derive D_s^{RKP} and D_s^{RKP-DE} as,

$$D_s^{RKP} = D_p \cdot P_{couple}, \tag{6}$$

$$D_s^{RKP-DE} = D_p \cdot P_{decouple}, \tag{7}$$

where recall that D_p denotes the average physical node degree. The improvement of D_s^{RKP-DE} over D_s^{RKP}, denoted by IM, is then given by,

$$IM = \frac{D_s^{RKP-DE} - D_s^{RKP}}{D_s^{RKP}} = \frac{P_3}{P_1 + P_2}. \tag{8}$$

Recall that P_1, P_2 and P_3 in (3), (4) and (5) respectively are functions of key pool size K, key chain size k and average physical node degree D_p. Under different values of D_p, we compute the values of D_s^{RKP}, D_s^{RKP-DE} and the improvement IM in Table I ($K = 10000$, $k = 50$). We can see that network decoupling improves the average secure node degree under all situations. The improvement in average secure node degree helps to enhance the performance of random key pre-distribution in terms of connectivity and resilience, which will be demonstrated using simulations in the following section.

TABLE I
IMPROVEMENT OF D_s^{RKP-DE} OVER D_s^{RKP} UNDER DIFFERENT D_p

D_p	5	10	15	20	25
D_s^{RKP}	1.66	4.26	7.59	11.52	15.89
D_s^{RKP-DE}	2.27	6.16	10.96	16.22	21.68
IM	37%	45%	44%	41%	36%

V. PERFORMANCE EVALUATION

In this section, we report experimental data to demonstrate the performance of our RKP-DE protocol compared to the traditional RKP protocol under various network and attack parameters. The metrics we study are connectivity (local connectivity and global connectivity) and resilience. We also study the overhead of our RKP-DE protocol compared to the traditional RKP protocol in terms of communication and computation.

A. Simulation Environment

The sensor network is a square region of size $1000m$ * $1000m$, in which 1000 sensors are deployed uniformly at random. The communication range r is the same for all sensors and is chosen based on the desired average physical node degree D_p (ignore the boundary effect). Each node is aware of the key sharing and node neighborhood information within its communication range. The following are the default values for the parameters unless otherwise specified: average physical node degree $D_p = 10$, key pool size $K = 10000$, key chain size $k = 50$. The attack model is one where the attacker can monitor all links in the network, and can capture up to x nodes. By default, $x = 50$. Each simulation is run 100 times with different random seeds, and the data presented is the average over 100 runs. In both RKP protocol and RKP-DE protocol, each node tries to establish a pair-wise key with each of its physical neighbors using multiple key paths based solely on its local information.

B. Sensitivity of Connectivity to D_p

1) Local Connectivity: In Fig. 7, we study the sensitivity of local connectivity to average physical node degree D_p. We observe that the local connectivity in RKP-DE protocol is consistently higher than that in the RKP protocol. The improvement is in fact more significant (about 35% improvement) for non-highly dense networks where $D_p < 20$. In RKP protocol, the consideration of both physical and logical constraints in key path construction limits the number of key paths between physical neighbors, especially when the network is not dense. However, the relaxation/decoupling of the constraints as a result of network decoupling enables the availability of many more key paths greatly enhancing local connectivity.

2) Global Connectivity: The definition of global connectivity here is the percent of nodes in the largest connected component of the secure network (an example is shown in Fig. 1 (b) or (c)). In Fig. 8, we observe that the global connectivity of our

[2] Not shown in Fig. 3 (b)

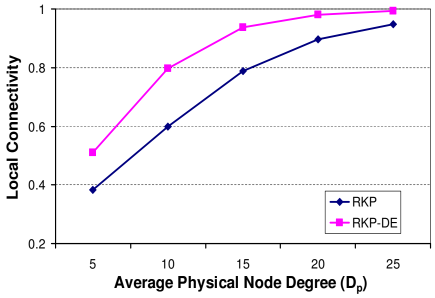


Fig. 7. Sensitivity of local connectivity to D_p

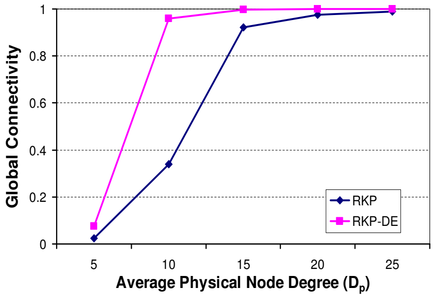


Fig. 8. Sensitivity of global connectivity to D_p

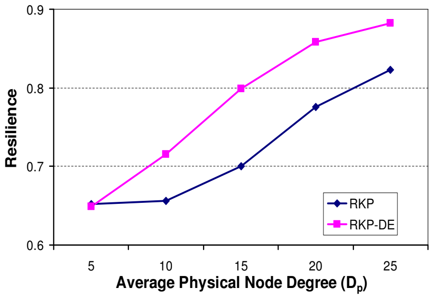


Fig. 9. Sensitivity of resilience to D_p

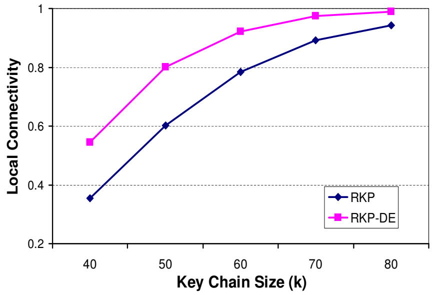


Fig. 10. Sensitivity of local connectivity to k

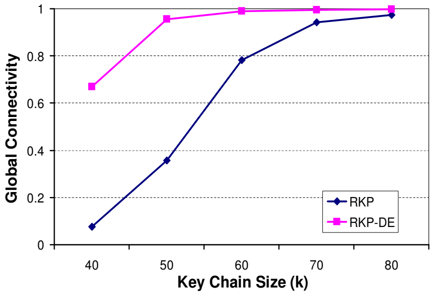


Fig. 11. Sensitivity of global connectivity to k

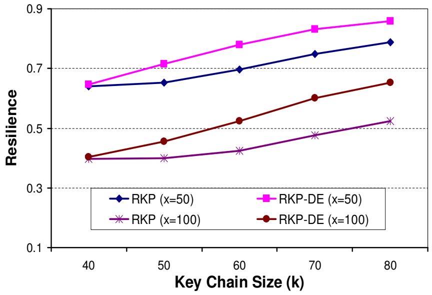


Fig. 12. Sensitivity of resilience to k and x

RKP-DE protocol is higher than that of the RKP protocol in all situations. The improvement is especially significant in non-highly dense networks (up to 200% improvement). This improvement is a result of the phase transition phenomenon in random graphs [21]. According to this phenomenon, the largest connected component in a random graph with n nodes jumps from $\Theta(\log n)$ to $\Theta(n)$ when the average node degree reaches beyond a certain threshold. With network decoupling in our RKP-DE protocol, such a jump in global connectivity occurs when D_p is around 10 compared to the RKP protocol when D_p is around 15. Another observation is that the global connectivity when $D_p = 10$ in our RKP-DE protocol is similar to the global connectivity when $D_p = 20$ in the RKP protocol. This demonstrates that we can obtain similar levels of global connectivity with much fewer nodes compared to the number of nodes needed in the RKP protocol.

C. Sensitivity of Resilience to D_p

In Fig. 9, we study the sensitivity of resilience to D_p. We see that the resilience is higher in our RKP-DE protocol compared to that of the traditional RKP protocol in general. The improvement is consistent except when the network is very sparse ($D_p = 5$). Network decoupling not only increases the number of key paths between physically neighboring nodes, but also decreases the number of logical hops of many key paths, both of which help enhance the resilience. When network becomes very sparse, only a single key path can be constructed for most situations, thus the improvement diminishes.

D. Sensitivity of Connectivity and Resilience to k and x

In Fig. 10, 11 and 12, we study the sensitivity of connectivity and resilience to k and x. In Fig. 10 and 11, we see similar pattern in sensitivity of connectivity to k as that to D_p. This is because the increase in k enhances the probability that two nodes share pre-distributed keys, which makes the local logical graph more dense. This can also be achieved by increasing D_p as well. Overall, our RKP-DE protocol achieves better performance than that of RKP protocol, and the performance improvement is especially significant in non-highly dense network. On the other hand, given the same performance requirement, our RKP-DE protocol can save storage overhead (k) up to around 30% compared with the RKP protocol. For example, given $k = 80$ in the RKP protocol, our RKP-DE protocol can achieve similar performance with k around (or smaller than) 60.

In Fig. 12, we study the sensitivity of resilience to key chain size k under different values for number of captured nodes x. We observe that the resilience of our RKP-DE protocol is better than that of the RKP protocol for all cases. The improvement is especially more pronounced for larger x (i.e., stronger attacks), which further demonstrates the effectiveness of our RKP-DE protocol. The value of x does not impact connectivity, so we do not show the sensitivity of connectivity to x.

E. Overhead

An important ancillary factor judging the performance of our protocol is the incurred overhead. The storage overhead (k) in our RKP-DE protocol is less than that of the RKP protocol under similar performance, as discussed above. Here we focus our discussion on communication and computation overhead.

1) Communication Overhead: In our protocol, each sensor establishes pair-wise keys with D_s secure neighbors on average. In order to establish a pair-wise key with each secure neighbor, the sensor needs to send a messages on each key path. If we denote the average number of key paths between a pair of sensors as n_p, and denote the average number of hops of a physical key path as h_p, the average number of messages a sensor sends/forwards is $\frac{D_s \cdot n_p \cdot h_p}{2}$. In practice, sensors may not need to use all the key paths available if the resilience requirement can be met with a few short physical key paths. Therefore, the values for n_p and h_p will be relatively small. Overall, the communication overhead in our protocol is similar to that of the traditional RKP protocol.

2) Computation Overhead: The computation overhead is dominated by two major parts in our protocol, which are key paths construction and key shares transmission. In key paths construction, the variant of depth-first-search algorithm we use is lightweight, especially in non-dense network where D_p is moderate. In key shares transmission, the encryption/decryption operation adopts a lightweight symmetric algorithm, and this operation occurs only on a logical hop basis. Overall, the computation overhead is mild.

VI. Final Remarks

In this paper, we proposed *network decoupling* to separate the logical relationship from the physical relationship in random key pre-distributed sensor networks. We designed a secure neighbor establishment protocol (RKP-DE) in decoupled sensor networks, and also designed a set of dependency elimination rules for eliminating link and path level key dependencies among the key paths. We conducted detailed analysis as well as extensive simulations to evaluate our proposed solution. Our data showed that significant performance improvement can be achieved using our solution in non-highly dense networks. Our future work will consist of practically implementing our proposed solution on the existing sensor network testbed at OSU [22].

Acknowledgment

We thank the anonymous reviewers for their invaluable feedback. This work was partially supported by NSF under grants No. ACI-0329155 and CCF-0546668.

References

[1] W. Diffie and M. E. Hellman, “New directions in cryptography,” *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644–654, November 1976.

[2] R. L. Rivest, A. Shamir, and L. M. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, February 1978.

[3] B. C. Neuman and T. Tso, “Kerberos: an authentication service for computer networks,” *IEEE Communications Magazine*, vol. 32, no. 9, pp. 33–38, September 1994.

[4] L. Eschenauer and V. D. Gligor, “A key-management scheme for distributed sensor networks,” in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, November 2002.

[5] H. Chan, A. Perrig, and D. Song, “Random key predistribution schemes for sensor networks,” in *Proceedings of IEEE Symposium on Research in Security and Privacy*, May 2003.

[6] A. Wacker, M. Knoll, T. Heiber, and K. Rothermel, “A new approach for establishing pairwise keys for securing wireless sensor networks,” in *Proceedings of the 3rd ACM Conference on Embedded Networked Sensor Systems (Sensys)*, November 2005.

[7] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, “A pairwise key predistribution scheme for wireless sensor networks,” in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, October 2003.

[8] D. Liu and P. Ning, “Establishing pairwise keys in distributed sensor networks,” in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, October 2003.

[9] S. Zhu, S. Xu, S. Setia, and S. Jajodia, “Establishing pairwise keys for secure communication in ad hoc networks: a probabilistic approach,” in *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP)*, November 2003.

[10] D. Verma, “Decoupling qos guarantees and connection establishment in communication networks,” in *Proceedings of Workshop on Resource Allocation Problems in Multimedia Systems (in conjunction with RTSS)*, December, 1996.

[11] A. Snoeren and B. Raghavan, “Decoupling policy from mechanism in internet routing,” in *Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Networking (HotNets-II)*, November 2003.

[12] H. Kung and S. Wang, “Tcp trunking: Design, implementation and performance,” in *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP)*, November, 1999.

[13] D. Niculescu and B. Nath, “Trajectory based forwarding and its applications,” in *Proceedings of the 9th ACM International Conference on Mobile Computing and Networking (MOBICOM)*, September 2003.

[14] J. Hwang and Y. Kim, “Revisiting random key pre-distribution schemes for wireless sensor networks,” in *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, October 2004.

[15] M. Miller and N. Vaidya, “Leveraging channel diversity for key establishment in wireless sensor networks,” in *Proceedings of the 25th IEEE Conference on Computer Communications (INFOCOM)*, April 2006.

[16] P. Traynor, H. Choi, G. Cao, S. Zhu, and T. L. Porta, “Establishing pair-wise keys in heterogeneous sensor networks,” in *Proceedings of the 25th IEEE Conference on Computer Communications (INFOCOM)*, April 2006.

[17] D. Huang, M. Mehta, D. Medhi, and L. Harn, “Location-aware key management scheme for wireless sensor networks,” in *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, October 2004.

[18] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney, “A key management scheme for wireless sensor networks using deployment knowledge,” in *Proceedings of the 23rd IEEE Conference on Computer Communications (INFOCOM)*, March 2004.

[19] D. Liu, P. Ning, and W. Du, “Group-based key pre-distribution in wireless sensor networks,” in *Proceedings of ACM Workshop on Wireless Security (WiSe)*, September 2005.

[20] W. Gu, X. Bai, S. Chellappan, and D. Xuan, “Network decoupling for secure communications in wireless sensor networks,” Dept. of CSE, The Ohio-State University, Columbus, OH, Tech. Rep. OSU-CISRC-3/06-TR27, March 2006.

[21] J. Spencer, *The Strange Logic of Random Graphs, Algorithms and Combinatorics 22*. Springer-Verlag, 2000.

[22] E. Ertin, A. Arora, R. Ramnath, and M. Nesterenko, “Kansei: A testbed for sensing at scale,” in *Proceedings of the 4th Symposium on Information Processing in Sensor Networks (IPSN/SPOTS track)*, April 2006.