

Network Decoupling: A Methodology for Secure Communications in Wireless Sensor Networks

Wenjun Gu, Xiaole Bai, Sriram Chellappan, *Student Member, IEEE*,
Dong Xuan, *Member, IEEE*, and Weijia Jia, *Member, IEEE*

Abstract—Many wireless sensor network (WSN) applications demand secure communications. The random key predistribution (*RKP*) protocol has been well accepted in achieving secure communications in WSNs. A host of key management protocols have been proposed based on the *RKP* protocol. However, due to the randomness in key distribution and strong constraint in key path construction, the *RKP*-based protocols 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 address this problem. With this methodology, a WSN is decoupled into a *logical keysharing network* and a *physical neighborhood network*, which significantly releases the constraint in key path construction of the *RKP* protocol. We design two new key management protocols, that is, *RKP-DE* and *RKP-DEA*, as well as a set of link and path dependency elimination rules in decoupled sensor networks. Our analytical and simulation data demonstrate the performance enhancement of our solutions from the perspective of connectivity and resilience and its applicability in nonhighly dense sensor networks.

Index Terms—Wireless sensor networks, random key predistribution, network decoupling.

1 INTRODUCTION

IN this paper, we address the issue of providing secure communications in wireless sensor networks (WSNs). WSNs are gaining wide acceptance today with a host of new applications being realized involving many tiny wireless sensors performing sensing and communication tasks. Many of these applications are in hostile/vulnerable environments, and their success is contingent on preventing the WSNs information from being accessible to external malicious attackers.

1.1 Motivation

In order to provide secure communications in WSNs, secret keys need to be established between communicating sensors. A host of key distribution techniques have been proposed for wired and wireless ad hoc networks. However, they cannot be applied in WSNs due to the unique characteristics of WSNs, like hostile zone deployment, ease of node capture, physical constraints in energy and memory, and so forth. For instance, public-key cryptography [1], [2] is too energy consuming for energy-constrained sensors. The key-distribution-center-based scheme [3] is centralized and not scalable when the network size increases. Using a single master key for all communications is too vulnerable, whereas establishing a unique pairwise key for each pair of nodes requires too

much memory. Also, when many sensors are to be deployed, random deployment may be the only choice. Since neighborhood information cannot be known in advance in such cases, predistributing pairwise keys between neighboring sensors prior to deployment is not possible too.

In order to address the above concerns, the *Random Key Predistribution (RKP)* protocol was first proposed in [4]. In the *RKP* protocol, each sensor is predistributed with k distinct keys randomly chosen from a key pool of K keys before deployment. Using the predistributed keys, two neighboring sensors attempt to establish a pairwise key for secure communications between themselves. We denote the neighbors of a sensor with which it can establish pairwise keys as *secure neighbors*. The *RKP* protocol has been well accepted in WSNs due to its good performance, distributed nature, simplicity, energy efficiency, and scalability. As such, it has served as a foundation for a host of key management protocols in WSNs [5], [6], [7], [8], [9], [10].

However, most *RKP*-based protocols have an inherent limitation in that the security performance is satisfactory only in highly dense networks, where the average number of physical neighbors per node (that is, average physical node degree) is large (≥ 20) [4], [5], [6]. As we know, such a high density is not always feasible in practice due to high deployment cost, increased chance of collisions, low per node throughput, etc. Due to the randomness in key distribution and strong constraint in the key path construction of *RKP* protocols (discussed subsequently), it often happens that the secure node degree (that is, the number of secure neighbors for a sensor) is very low in nonhighly dense networks. Consequently, the networks will have low secure connectivity and are very likely to be partitioned as illustrated in Fig. 1. The original sensor network is shown in Fig. 1a. There is an edge between two nodes if they are physical neighbors. The average physical node degree is

• W. Gu, X. Bai, S. Chellappan, and D. Xuan are with the Department of Computer Science and Engineering, The Ohio State University, DL395, 2015 Neil Avenue, Columbus, OH 43210.

E-mail: {gu, baixia, chellapp, xuan}@cse.ohio-state.edu.

• W. Jia is with the Department of Computer Science, City University of Hong Kong, 83 Tat Chee Ave., Kowloon, Hong Kong, SAR China.

E-mail: wjia@cs.cityu.edu.hk.

Manuscript received 21 May 2006; revised 3 Jan. 2007; accepted 7 Feb. 2007; published online 1 Mar. 2007.

Recommended for acceptance by C. Raghavendra.

For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0131-0506.

Digital Object Identifier no. 10.1109/TPDS.2007.1078.

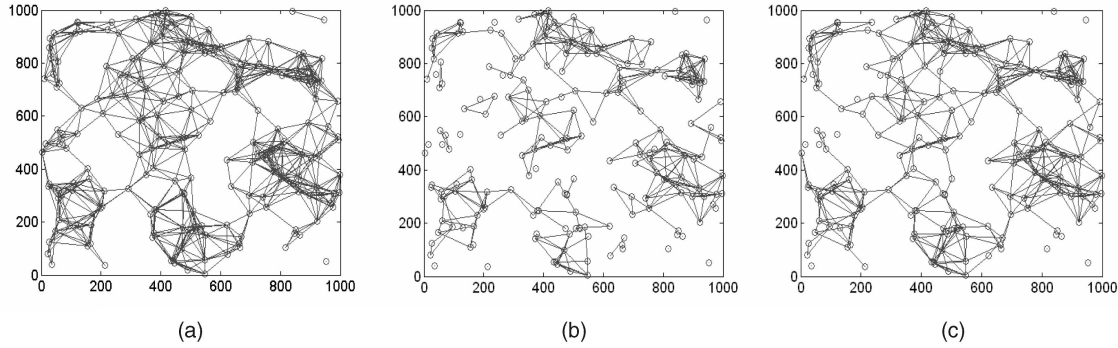


Fig. 1. Average secure node degree comparison between *RKP* and *RKP-DE*. Our *RKP-DE* achieves 40 percent improvement in the average secure node degree. The network is of size $1,000\text{ m} \times 1,000\text{ m}$, where 200 nodes are deployed uniformly at random. All nodes have the same communication range (133 m), and the average physical node degree is 9.71. We set $K = 10,000$ and $k = 50$. (a) Original network. (b) Network with *RKP*. (c) Network with *RKP-DE*.

9.71. The corresponding secure network as a result of executing the *RKP* protocol is shown in Fig. 1b, where an edge exists between two nodes if they are secure neighbors. In this example, we limit the number of intermediate nodes on a key path to be one.¹ The average secure node degree in Fig. 1b is only 4.06. It is much smaller compared to the average physical node degree. As can be seen, the network in Fig. 1b is partitioned into many components. Two nodes cannot communicate securely if they reside in different components.

1.2 Our Contributions

In this paper, we address the above problem. Our contributions are fourfold:

- *Network decoupling methodology.* We propose a methodology called *network decoupling* for secure communications in WSNs. In network decoupling, we decouple the logical keysharing relationship from the physical neighborhood relationship in sensor networks. The flexibility offered by decoupling greatly enhances the chances of pairwise key establishment between physical neighbors.
- *Protocol design.* We design two new key management protocols for secure neighbor establishment in decoupled sensor networks. In our first protocol called *RKP-DE*, logical key paths are constructed based on the logical keysharing relationship, and then, the corresponding physical key paths are constructed based on the physical neighborhood relationship. We then design an adaptive protocol called *RKP-DEA*, which takes network heterogeneity into consideration during protocol execution. This protocol is particularly well suited for achieving high-security performance in heterogeneous sensor networks.
- *Dependency elimination rules.* We propose novel dependency elimination rules in our protocols to detect and eliminate link and path dependencies without compromising resilience. In pairwise key establishment, when multiple key paths are constructed, some links (or paths) could be dependent

on other links (or paths). Such dependencies introduce unnecessary overhead in terms of communication and computation, which can be eliminated using our proposed rules.

- *Analysis.* We conduct a formal analysis of our protocols from the perspective of the average secure node degree and overhead. The communication overhead and computation overhead are studied by analyzing a new metric called *stretch factor* and the algorithm complexity, respectively. Our analysis demonstrates a significant improvement in the average secure node degree with a mild increase in overhead in our protocols compared to the *RKP* protocol.

To illustrate the performance improvement of the network decoupling methodology, we show the corresponding secure network as a result of executing the *RKP-DE* protocol in Fig. 1c (under the same configuration). The average secure node degree in Fig. 1c has now increased to 5.68, a 40 percent improvement over that in Fig. 1b. The extensive analysis and simulations conducted in this paper further validate this fact.

The rest of our paper is organized as follows: We discuss the traditional *RKP* protocol in Section 2. The methodology of network decoupling is introduced in Section 3. We discuss our basic *RKP-DE* protocol and adaptive *RKP-DEA* protocol in Sections 4 and 5, respectively. We then present the analysis and performance evaluations in Sections 6 and 7, respectively. After discussing related work in Section 8, we finally conclude our paper in Section 9.

2 THE RANDOM KEY PREDISTRIBUTION PROTOCOL IN WSNs

In this section, we give a brief overview of the *RKP* protocol. There are two stages in this protocol [4]: *key predistribution* and *secure neighbor establishment*. In the *key predistribution* stage, each node is predistributed with k distinct keys randomly chosen from a large key pool of K keys, and then, the nodes are deployed randomly in the network. The set of keys predistributed in a node is called its *key chain*.

1. The general case of multiple intermediate nodes on a key path will be discussed later.

Once nodes are deployed, the *secure neighbor establishment* stage follows. First each node sends a message to its physical neighbors, containing its node ID and the key IDs of its predistributed keys. Then, two physical neighbor nodes attempt to establish a pairwise key via an existing secure communication between them. Here, secure communication is defined as the communication between two nodes where all messages transmitted (possibly via multi-hops) are encrypted. If two physical neighbor nodes share at least one predistributed key, they can establish a pairwise key directly. Otherwise, two physical neighbor nodes may still be able to establish a pairwise key via the help of other nodes called *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 predistributed key, and the pairwise key is encrypted/decrypted in *each* hop till it reaches the destination.

The standard attack model is one where the attacker attempts to decipher as much information as possible from sensor communications [4], [5], [7]. The attacker has the ability to monitor and record all the wireless communication in the network *immediately* after node deployment (that is, link monitor attack). Besides, the attacker is assumed to be able to physically capture a limited number of nodes in the network (that is, node capture attack). Once a node is captured, its predistributed keys and pairwise keys are all disclosed to the attacker.

To evaluate the performance of the *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 physical neighbor nodes are able to establish a pairwise key between them. Global connectivity is defined as either the probability that the whole secure network (for example, Figs. 1b or 1c) 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 pairwise 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.

3 NETWORK DECOUPLING IN RANDOM KEY PREDISTRIBUTED SENSOR NETWORKS

3.1 Network Decoupling

In random key predistributed sensor networks, there exist two types of relationship between any two nodes. One is logical (sharing predistributed keys), and the other is physical (within communication range). We can separate these two types of relationship by decoupling a random key predistributed 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 predistributed key. Similarly, two nodes in the physical graph have an edge between them if they are within the communication range of each other. For the example in Fig. 2a, its decoupled logical and physical graphs are shown in Figs. 2b and 2c, respectively.

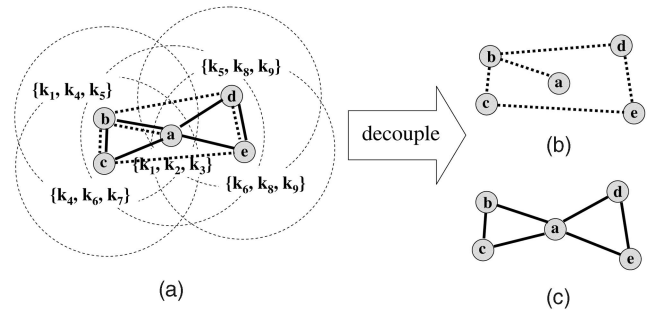


Fig. 2. Decouple a sensor network into a logical graph and a physical graph. (a) Sample sensor network. (b) Logical graph. (c) Physical graph.

The detailed description on how nodes construct these graphs is presented in Section 4.2.

Now, we will show how network decoupling achieves secure communication. There are two cases possible where two nodes in the network can communicate securely. The first case is where two nodes share at least one predistributed key (that is, they are *directly* connected in the logical graph), and the nodes are connected in the physical graph. In this case, the source node can encrypt the messages using the shared key, and each intermediate node in the physical graph simply forwards the messages toward the destination. Such intermediate nodes in the physical graph are called *physical intermediate nodes*. An example of the first case is nodes *b* and *d* in Fig. 2a, where node *a* is the physical intermediate node between them. The second case is one where two nodes do not share a key but are connected *indirectly* in the logical graph with multiple logical hops, and the two nodes for each logical hop are connected in the physical graph. In this case, encryption occurs at each intermediate node in the logical graph, whereas each intermediate node in the physical graph simply forwards the messages. Such intermediate nodes in the logical graph are called *logical intermediate nodes*. An example of the second case is nodes *a* and *d* in Fig. 2a, where node *b* is the logical intermediate node between them.

3.2 Benefit of Network Decoupling

In this section, we will demonstrate the benefits of network decoupling by analysis. Specifically, we will derive the probability that two physical neighbor nodes are able to establish a pairwise key.

Formally, the probabilities that two physical neighbor nodes are able to establish a pairwise key in the *RKP* protocol (with multiple proxies), denoted by P_{couple} , and the *RKP-DE* protocol (with multiple logical intermediate nodes), denoted by $P_{decouple}$, are given by

$$P_{couple} = 1 - (1 - P_{couple}(A_a)) \cdot (1 - P_{couple}(A_b|\bar{A}_a)), \quad (1)$$

$$P_{decouple} = 1 - (1 - P_{decouple}(A_a)) \cdot (1 - P_{decouple}(A_b|\bar{A}_a)). \quad (2)$$

In the above expressions, $P_{couple}(A_a)$ and $P_{decouple}(A_a)$ denote the probability that node *a* is able to construct a key path to its physical neighbor node *b* based on local information within its information area in the *RKP* and *RKP-DE* protocols, respectively. By default, the information area is the communication range with size $A_a = \pi r^2$, where *r* is the communication range. In the above

equations, $P_{couple}(A_b|\bar{A}_a)$ and $P_{decouple}(A_b|\bar{A}_a)$ denote the probability that node b is able to construct a key path to node a based on its local information (within range $A_b = \pi r^2$), given that node a cannot construct a key path to node b within A_a , in the *RKP* and *RKP-DE* protocols, respectively. In the following, we will derive the expressions for $P_{decouple}(A_a)$ and $P_{decouple}(A_b|\bar{A}_a)$ first, followed by the derivation for $P_{couple}(A_a)$ and $P_{couple}(A_b|\bar{A}_a)$.

We first define some notations. We define $P_h(i)$ as the probability that a node a can find a logical key path to a physical neighbor node b in our *RKP-DE* protocol within the information area of node a with minimum logical hops i . Similarly, $P_h(i|A')$ is defined as the probability that node a can find a logical key path to node b in the *RKP-DE* protocol within the overlapped information area of nodes a and b with minimum logical hops i . We define $P_{decouple}(A')$ as the probability that node a is able to find a key path to node b within the overlapped information areas of nodes a and b ($A' = 0.5865 \cdot \pi r^2$ [5]) and define $P_{decouple}(A')$ as $1 - P_{decouple}(A')$. $P_{decouple}(A_a|A')$ and $P_{decouple}(A_a|\bar{A}')$ are conditional probabilities defined similarly as $P_{decouple}(A_b|\bar{A}_a)$ above. By the law of total probability, we have

$$P_{decouple}(A_a) = P_{decouple}(A') \cdot P_{decouple}(A_a|A') + P_{decouple}(\bar{A}') \cdot P_{decouple}(A_a|\bar{A}'). \quad (3)$$

In the above equation, $P_{decouple}(A_a)$ is given by $P_{decouple}(A_a) = \sum_{i=1}^{\infty} P_h(i)$. Similarly, $P_{decouple}(A')$ is given by $P_{decouple}(A') = \sum_{i=1}^{\infty} P_h(i|A')$. The value of $P_{decouple}(A_a|A')$ is always one since area A' is within area A_a . Therefore, we can obtain the expression of $P_{decouple}(A_a|\bar{A}')$ by

$$P_{decouple}(A_a|\bar{A}') = (P_{decouple}(A_a) - P_{decouple}(A')) / P_{decouple}(\bar{A}') = \left(\sum_{i=1}^{\infty} P_h(i) - \sum_{i=1}^{\infty} P_h(i|A') \right) / \left(1 - \sum_{i=1}^{\infty} P_h(i|A') \right). \quad (4)$$

Note that $P_{decouple}(A_b|\bar{A}_a)$ equals $P_{decouple}(A_a|\bar{A}')$. Finally, we have the expression of $P_{decouple}$:

$$P_{decouple} = 1 - (1 - P_{decouple}(A_a)) \cdot (1 - P_{decouple}(A_b|\bar{A}_a)) = 1 - (1 - \sum_{i=1}^{\infty} P_h(i)) \cdot \left(1 - \left(\sum_{i=1}^{\infty} P_h(i) - \sum_{i=1}^{\infty} P_h(i|A') \right) / \left(1 - \sum_{i=1}^{\infty} P_h(i|A') \right) \right). \quad (5)$$

The detailed derivation of $P_h(i)$ and $P_h(i|A')$ is given in the Appendix.

The analysis of the *RKP* protocol is similar to that of the *RKP-DE* protocol above. If we denote $P'_h(i)$ as the probability that node a can find a logical key path to a physical neighbor node b in the *RKP* protocol within the information area of node a with minimum logical hops i and denote $P'_h(i|A')$ as the probability that node a can find a

logical key path to node b in *RKP* protocol within the overlapped information areas of nodes a and b with minimum logical hops i , the expression of P_{couple} is given by

$$P_{couple} = 1 - (1 - P_{couple}(A_a)) \cdot (1 - P_{couple}(A_b|\bar{A}_a)) = 1 - \left(1 - \sum_{i=1}^{\infty} P'_h(i) \right) \cdot \left(1 - \left(\sum_{i=1}^{\infty} P'_h(i) - \sum_{i=1}^{\infty} P'_h(i|A') \right) / \left(1 - \sum_{i=1}^{\infty} P'_h(i|A') \right) \right). \quad (6)$$

Due to space limitation, we skip the derivation of $P'_h(i)$ and $P'_h(i|A')$. Interested readers are referred to [11] for details.

Based on our analysis in the Appendix, we will see that $P_h(i)$ is larger than $P'_h(i)$, $P_h(i|A')$ is larger than $P'_h(i|A')$, and $P_h(i) - P_h(i|A')$ is larger than $P'_h(i) - P'_h(i|A')$ for all i larger than 1 (corresponding parameters are equal when $i = 1$). By observing (5) and (6), we can see that $P_{decouple}$ is always larger than P_{couple} , which shows the benefit of our network decoupling.

4 SECURE NEIGHBOR ESTABLISHMENT PROTOCOL IN DECOUPLED NETWORKS

4.1 Overview

In this section, we discuss the design of our *RKP-DE* protocol for establishing secure neighbors (that is, establishing pairwise keys) in decoupled random key predistributed sensor networks. 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 key paths between neighboring nodes,
3. eliminating dependencies among the key paths, and
4. establishing pairwise keys between neighboring nodes.

The major differences between our *RKP-DE* protocol and the traditional *RKP* protocol are in the first three components. In the following, we will describe each component in our *RKP-DE* protocol in detail.

4.2 Local Graphs Construction

After node deployment, each node obtains the keysharing and physical neighborhood information in its information area by local communication with its physical neighbors. By sending the neighboring nodes' IDs to the neighbors, each node is aware of the fact whether two of its neighbors are within communication range or not. With this information, each node constructs a local logical graph (G_l) and a local physical graph (G_p). In the local logical graph (for example, Fig. 2b), two nodes are connected if they share at least one key, whereas in the local physical graph (for example, Fig. 2c), two nodes are connected if they are within communication range of each other.

4.3 Key Paths Construction

Algorithm 1 shows the pseudocode of key paths construction executed by node u . Initially, the logical key path tree (T_u) consists of a single vertex u . The key paths construction

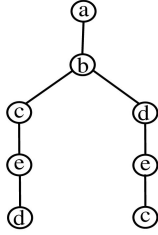


Fig. 3. Logical key path tree of node a .

is executed in two steps. First, T_u is constructed by node u based on its local logical graph $G_l(u)$ (lines 1 to 7), and T_u contains all the logical key paths between u and its secure neighbors. Then, node u constructs the 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 lines 3 and 11 will be discussed in Section 4.4. Here, $N(u)$ denotes the set of physical neighbors of node u .

Algorithm 1. Pseudocode of key paths construction

- 1: **Logical_Key_Path_Tree_Construction**($u, G_l(u), T_u$)
- 2: **for each** $v \in G_l(u)$
- 3: **if** $Link_Dependency_Checking(v, u, T_u) == PASS$, **then**
- 4: $Insert(u, v, T_u)$;
- 5: $Logical_Key_Path_Tree_Construction(v, G_l(u), T_u)$;
- 6: **endif**
- 7: **endfor**

- 8: **Physical_Key_Paths_Construction**($u, G_p(u), T_u$)
- 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: **endfor**

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

Logical key path tree construction. The protocol constructs a 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). Fig. 3 shows the resultant logical key path tree for node a in the example shown in Fig. 2b. By executing the algorithm just once on its local logical graph in Fig. 2b, 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 , which are $\langle a, b, c, e \rangle$ and $\langle a, b, d, e \rangle$.

Physical key paths construction. After obtaining the logical key path tree (T_u), node u begins to construct physical key paths (lines 8 to 13). For each neighbor v , node u first obtains a set of logical key paths between u and v (T_{uv}) from T_u . Out of all paths in T_{uv} , the set of paths passing the dependency checking is denoted as T'_{uv} . For all logical key paths in T'_{uv} , the corresponding physical key paths T^*_{uv} are obtained. In Fig. 2b, the logical key path $\langle a, b, d, e \rangle$ contains a logical hop $\langle b, d \rangle$ between two nonneighboring nodes. In Fig. 2c, we see that a physical path $\langle b, a, d \rangle$

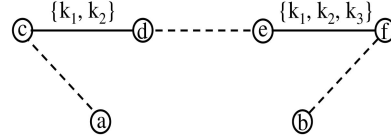


Fig. 4. Link dependency example.

can replace the above logical hop. Therefore, the corresponding physical key path is $\langle a, b, a, d, e \rangle$. Message encryption/decryption occurs for each logical hop, whereas message transmission occurs for each physical hop.

4.4 Dependency Elimination

We now discuss the elimination of link and path dependencies in lines 3 and 11 of Algorithm 1. Generally, if more key paths are used, resilience is enhanced. This is because the attacker needs to compromise all key paths in order to compromise the established pairwise key. 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. In this section, we propose two novel *dependency elimination rules* to decrease such overheads without affecting the resilience of the established pairwise keys.

4.4.1 Link Dependency Elimination

We illustrate link dependency with an example in Fig. 4. Node a obtains a logical key path $\langle a, \dots, c, d, \dots, e, f, \dots, b \rangle$ to a physical neighbor node b . We denote $K(i, j)$ as the set of keys shared by nodes i and j . There exists a link dependency between the hops $\langle c, d \rangle$ and $\langle e, f \rangle$ 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 $\langle a, \dots, c, f, \dots, b \rangle$, which 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, 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. Besides, using a shorter key path will save overhead. We formally define link dependency below.

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

Once a link dependency is detected, our *link dependency elimination rule* will eliminate the logical key path it resides. The pseudocode of link dependency checking is given in Algorithm 2 (lines 1 to 5). 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 $v.par$ denotes the parent node of node v on the tree T .

Algorithm 2. Pseudocode of dependency checking

- 1: **Link_Dependency_Checking**(v, u, T)
- 2: **if** \exists node $w \in Path(u, root)$, s.t.
 $K(v, v.par) \subseteq K(w, w.par)$ OR $K(w, w.par) \subseteq K(v, v.par)$,
then

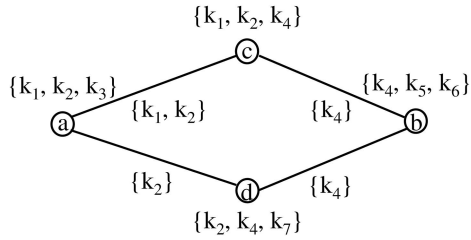


Fig. 5. Path dependency example.

```

3: return FAIL;
4: else return PASS;
5: endif

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

```

4.4.2 Path Dependency Elimination

Apart from link dependency, another type called path dependency may exist. In Fig. 5, there are two logical key paths between nodes a and b . However, we can see that the compromise of the key path $\langle a, c, b \rangle$ (disclosure of keys (k_1, k_2) , or (k_4)) always leads to the compromise of the other key path $\langle a, d, b \rangle$, but not vice versa. Therefore, given that key path $\langle a, c, b \rangle$ exists, the other key path $\langle a, d, b \rangle$ becomes redundant and also incurs unnecessary overhead. Denoting the set of logical hops on a logical key path p as L_p and denoting the set of keys used on a logical hop h as $K(h)$, path dependency is formally defined as follows.

Path dependency. Given two logical key paths p and q , there exists a 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' ($h' \in L_p$), such that $K(h') \subseteq K(h)$, and 2) \forall logical hop $h \in L_p$, \exists a logical hop h' ($h' \in L_q$), such that $K(h') \subseteq K(h)$.

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, the weaker one will be eliminated. The pseudocode of path dependency checking is given in Algorithm 2 (lines 6 to 14).

4.5 Pairwise Key Establishment

Once key paths are constructed, each sensor generates random keyshares and sends each keyshare on each physical key path. The messages are transmitted at each physical hop and are encrypted/decrypted at each logical hop. Take the logical key path $\langle a, b, d \rangle$ in Fig. 2b as an example. Its corresponding physical key path is $\langle a, b, a, d \rangle$. We assume

that a keyshare $k_a^{(1)}$ is transmitted on this key path. We denote $\{M\}_k$ as a message M encrypted with key k . The transmission of $k_a^{(1)}$ is executed as follows:

$$\begin{aligned}
a \rightarrow b &: \langle 1 \rangle, \{ \langle a \rangle, \langle a, d, 5 \rangle, \langle k_a^{(1)} \rangle \}_{k_1}, \\
b \rightarrow a &: \langle d \rangle, \langle 5 \rangle, \{ \langle a \rangle, \langle \phi \rangle, \langle k_a^{(1)} \rangle \}_{k_5}, \\
a \rightarrow d &: \langle 5 \rangle, \{ \langle a \rangle, \langle \phi \rangle, \langle k_a^{(1)} \rangle \}_{k_5}.
\end{aligned}$$

In the message that node a sends to node b , $\langle 1 \rangle$ denotes the ID of the key used to encrypt the remaining message, $\langle a \rangle$ denotes the source node, and $\langle a, d, 5 \rangle$ denotes the remaining physical key path and the ID of the key used to encrypt the message forwarded to the next node d on the logical key path.² In the message that b sends to a , $\langle d \rangle$ denotes the remaining physical path of the current logical hop since a cannot decrypt the message using key k_5 .

Similarly, node a can transmit another random keyshare $k_a^{(2)}$ on another logical key path $\langle a, b, c, e, d \rangle$ to node d . Node d may also construct other key paths (not shown in Fig. 2b) and transmit its keyshares to node a . Finally, nodes a and d can compute a common pairwise key via some simple operation such as a bitwise XOR operation, based on all the keyshares they both generated. In this way, the established pairwise key is compromised if and only if all the keyshares (key paths) are compromised.

5 ADAPTIVE SECURE NEIGHBOR ESTABLISHMENT PROTOCOL

5.1 Motivation

In our *RKP-DE* protocol discussed in Section 4, all nodes execute the protocol in a homogeneous way. That is, all nodes have the *same* information area (one-hop communication range), and all nodes construct *all* key paths available for pairwise key establishment similar to each other. However, in reality, it may happen that despite the random deployment of nodes and pre-distributed keys, there will be heterogeneity in the network. This heterogeneity can be in terms of the number of physical neighbors per nodes and the chances of nodes sharing pre-distributed keys with their neighbors. In such cases, a homogeneous protocol may not achieve a satisfactory performance. Let us denote *poor* nodes as those that cannot construct key paths and establish pairwise keys with most of their neighbors even if they utilize all the key paths available and denote *rich* nodes as those that can construct quite a few key paths to most of their neighbors. In such cases, poor nodes need to find more key paths by increasing their information areas (enhancing connectivity and resilience), whereas rich nodes may not need to utilize all key paths (decreasing overhead). Our adaptive *RKP-DEA* protocol achieves this objective.

Same as the *RKP-DE* protocol, our *RKP-DEA* protocol also has four components: local graphs construction, key paths construction, dependency elimination, and pairwise key establishment. We propose adaptive mechanisms for the first two components, which will be discussed in detail below.

2. The next node on the logical key path is the node before the first number in the list.

5.2 Adaptive Local Graphs Construction

In the *RKP-DE* protocol, the size of the information area is the same for all nodes. As discussed above, local node density and keysharing can vary across the network. Therefore, the size of information area should be decided by each node based on its local node density and local keysharing, the details of which are discussed below. After the information area size is decided, each node exchanges information with the other nodes in its information area and constructs local logical/physical graphs based on the obtained information in the same way as it does in our *RKP-DE* protocol discussed in Section 4.2.

After node deployment, each node has an initial information area of its one-hop communication range. Each node constructs key paths to its neighbors based on the node information and keysharing information in its initial information area. Since there is no guarantee that a pairwise key can be established between any two neighboring nodes due to the randomness of key predistribution, we need a threshold ($THRESH_SN$), which denotes the required percent of neighbors with which a pairwise key can be established (for example, the percentage of secure neighbors). The value of $THRESH_SN$ can be decided based on the required probability that the whole secure network is connected (P_c) and the total number of nodes (N) by random graph theory in [12]:

$$THRESH_SN = \frac{\ln N - \ln(-\ln P_c)}{N}. \quad (7)$$

If a node s can achieve the above threshold, its information area will keep the same. Otherwise, node s will increase its information area to be its two-hop communication range. Besides, node s will request its current nonsecure neighbors to increase their information areas by one more hop as well. In this way, node s will have a higher chance of finding key paths between itself and its nonsecure neighbors. Both node s and its nonsecure neighbors obtain extra information in their new information areas and construct key paths based on updated information. After this, node s will check whether the threshold is achieved or not. Depending on the result of the check, node s will decide whether to stop increasing its information area or increase it further by one more hop. In order to prevent message flooding incurred by a few poor nodes that can never achieve the above threshold, we need an upper bound of information area size $MAX_INFOAREA$, which is the maximum hop number of information area. Having a few nodes not achieving the above threshold will not pose a major impact to the global network security performance. The value of $MAX_INFOAREA$ can be decided by our analysis in Section 3. Given the information area size A , we can obtain the percentage of secure neighbors $P_{decouple}(A)$ in (5). The value of $MAX_INFOAREA$ can be set as twice the minimum h such that $P_{decouple}(\pi(hr)^2) \geq THRESH_SN$, where $THRESH_SN$ is given in (7).

5.3 Adaptive Key Paths Construction

In the *RKP-DE* protocol, each node constructs all key paths available for pairwise key establishment. As we point out above, establishing a pairwise key via all the key paths

could be an overkill when the number of available key paths is relatively large. In our adaptive key paths construction, we allow each node to select a subset of key paths for pairwise key establishment.

The main difference between our adaptive key paths construction and the basic key paths construction in Algorithm 1 in Section 4 is that we will choose a subset of the logical key paths, denoted by $T_{uv-adaptive}$, instead of T_{uv} for physical key paths construction. We first compute the path resilience (discussed below) for all logical key paths in T_{uv} . Then, we sort the logical key paths based on path resilience in descending order and add one logical key path into the selected subset of key paths $T_{uv-adaptive}$ (initially empty) each time until either all logical key paths are chosen or the key resilience (discussed below) based on the currently selected subset of logical key paths satisfies the required key resilience threshold $THRESH_RES$. The value of $THRESH_RES$ can be decided by the security requirement of the application. In the following, we will discuss how to derive path resilience and key resilience.

We first derive hop resilience $R_{hop}(k_i)$, which is defined as the probability that a logical hop with k_i shared keys is not compromised. The expression of $R_{hop}(k_i)$ is given by

$$R_{hop}(k_i) = 1 - \binom{K - k_i}{k_{dis} - k_i} / \binom{K}{k_{dis}}, \quad (8)$$

where K is the key pool size, and k_{dis} is the average number of disclosed keys given that x nodes are captured. This is because a logical hop with k_i shared keys is compromised if all k_i keys are disclosed. The expression for k_{dis} is given by $k_{dis} = K \cdot (1 - (1 - \frac{x}{K})^x)$.

For a logical key path P with h logical hops, we denote the number of shared keys on each logical hop by k_i ($1 \leq i \leq h$) and denote the path resilience of path P by $R_{path}(P)$. The expression for $R_{path}(P)$ is given by

$$R_{path}(P) = \prod_{i=1}^h R_{hop}(k_i). \quad (9)$$

This is because a logical key path is uncompromised if all the logical hops are uncompromised.

For a pairwise key established by a set of logical key paths (P_1, P_2, \dots, P_m) , its resilience, denoted by R_{key} , can be given by

$$R_{key} = 1 - \prod_{i=1}^m (1 - R_{path}(P_i)). \quad (10)$$

This is because a pairwise key is uncompromised if at least one logical key path is uncompromised.

The computation of path and key resilience above is based on the assumption that shared keys on different logical hops in the same or different key path(s) are independent. The computation of path and key resilience considering the link and path dependency is much more complicated and computationally expensive for energy-constrained sensors. Our approximation above is simple but effective. Besides, in the derivation of k_{dis} , we need to know the number of captured nodes x . The exact value of x may not be known during our protocol execution; however, we can estimate x based on historical data.

TABLE 1

Comparison of the Average Secure Node Degree in the *RKP* Protocol and *RKP-DE* Protocol under Different D_p

D_p	5	10	15	20	25
$D_s^{RKP(1)}$	1.66	4.26	7.59	11.52	15.89
$D_s^{RKP-DE(1)}$	2.27	6.16	10.96	16.22	21.68
$IM^{(1)}$	37%	45%	44%	41%	36%
D_s^{RKP}	1.93	6.07	11.62	17.75	23.80
D_s^{RKP-DE}	2.63	8.05	14.43	19.85	24.98
IM	37%	33%	24%	12%	5%

6 ANALYSIS

6.1 Average Secure Node Degree

In this section, we will derive the expressions for the average secure node degree in both *RKP* and *RKP-DE* protocols, denoted by D_s^{RKP} and D_s^{RKP-DE} , respectively. In Section 3, we gave the expressions for P_{couple} and $P_{decouple}$ in (6) and (5), respectively. Therefore, we can derive D_s^{RKP} and D_s^{RKP-DE} as

$$D_s^{RKP} = D_p \cdot P_{couple}, \quad (11)$$

$$D_s^{RKP-DE} = D_p \cdot P_{decouple}. \quad (12)$$

The improvement of D_s^{RKP-DE} over D_s^{RKP} , denoted by IM , is then given by

$$IM = \frac{D_s^{RKP-DE} - D_s^{RKP}}{D_s^{RKP}}. \quad (13)$$

We compute the values of D_s^{RKP} , D_s^{RKP-DE} , and IM under different D_p in Table 1 ($K = 10,000$, $k = 50$). In Table 1, we also give the values of $D_s^{RKP(1)}$, $D_s^{RKP-DE(1)}$, and $IM^{(1)}$, which are the average secure node degree by using only key paths consisting of one proxy in *RKP* and one logical intermediate node in *RKP-DE*, and the improvement of $D_s^{RKP-DE(1)}$ over $D_s^{RKP(1)}$, respectively. The expressions for $D_s^{RKP(1)}$ and $D_s^{RKP-DE(1)}$ are the same as (11) and (12), except that we replace P_{couple} and $P_{decouple}$ by $P_{couple}^{(1)}$ and $P_{decouple}^{(1)}$ (derived in [13]), respectively. We can see that network decoupling improves the average secure node degree under all situations, which helps to enhance the performance of *RKP* in terms of connectivity and resilience. We also observe that the improvement in case of multiple logical intermediate nodes (IM) diminishes for a larger D_p . This is because in a highly dense network, most physical neighbor nodes are able to establish pairwise keys. Therefore, the value of D_s^{RKP} is close to that of D_p , and the improvement diminishes.

In the above, we show the derivation of the average secure node degree of our *RKP-DE* protocol. Compared with the *RKP-DE* protocol, our *RKP-DEA* protocol can achieve an even higher average secure node degree. This is because we allow poor nodes to increase their information areas so that they can establish pairwise keys with more neighbors.

TABLE 2

Comparison of the Stretch Factor in the *RKP* Protocol and *RKP-DE* Protocol under Different D_p

D_p	5	10	15	20	25
SF^{RKP}	1.54	1.91	2.01	2.04	2.03
SF^{RKP-DE}	2.14	2.53	2.46	2.37	2.30

6.2 Stretch Factor

In this paper, we define a new metric called *stretch factor* to study the protocol communication overhead. Formally, the stretch factor is the average number of physical hops on the key path between two secure neighbors. It reflects the communication overhead of a protocol since a message needs to be transmitted/forwarded once for each physical hop during keyshares transmission, which dominates the other components in the protocol in terms of communication overhead. We denote the stretch factor for the *RKP* protocol and *RKP-DE* protocol as SF^{RKP} and SF^{RKP-DE} , respectively, and derive them below.

Let us denote $P_h(i)$ and $P_h(i)'$ as the probability that a node can find a logical key path to a physical neighbor node within its information area with minimum logical hop i in the *RKP-DE* protocol and *RKP* protocol, respectively. We further denote α as the average number of physical hops for a logical hop on a key path (except the first logical hop). Then, SF^{RKP} and SF^{RKP-DE} are given by

$$SF^{RKP} = \sum_{i=1}^{\infty} \left((1 + (i-1)\alpha) \cdot P_h'(i) \right) / \sum_{i=1}^{\infty} (P_h'(i)), \quad (14)$$

$$SF^{RKP-DE} = \sum_{i=1}^{\infty} \left((1 + (i-1)\alpha) \cdot P_h(i) \right) / \sum_{i=1}^{\infty} (P_h(i)). \quad (15)$$

In the above equations, $1 + (i-1)\alpha$ denotes the average physical hops of a key path with i logical hops. This is because the first logical hop is between the source node and one of its physical neighbors (resulting in one physical hop). For each of the remaining $i-1$ logical hops, the two nodes of that logical hop are within communication range (one physical hop) with probability 0.5865 [5] and are connected by the source node (two physical hops) with probability $1 - 0.5865 = 0.4135$. Therefore, each of the remaining $i-1$ logical hops has the average number of physical hops $\alpha = 1 \cdot 0.5865 + 2 \cdot 0.4135 = 1.4135$. The derivations of $P_h(i)$ and $P_h(i)'$ are given in the Appendix.

In Table 2, we show the values of SF^{RKP} and SF^{RKP-DE} under various D_p for $K = 10,000$ and $k = 50$. We can see that the stretch factor in the *RKP-DE* protocol is only slightly larger than that of the *RKP* protocol for the same D_p , which is due to the dual effects of network decoupling discussed below. With network decoupling, our *RKP-DE* protocol can construct more key paths than the *RKP* protocol. Therefore, we can partition the set of key paths constructed by the *RKP-DE* protocol (T^*) into two disjoint subsets T_1^* and T_2^* in which T_1^* denotes the set of key paths constructed by both *RKP* and *RKP-DE* protocols, and T_2^* denotes the set of key paths constructed by only the

RKP-DE protocol. With network decoupling, the stretch factor of the key paths in T_1^* is decreased. However, the stretch factor of key paths in T_2^* is larger than that of key paths in T_1^* . The result of the dual effects above is that the stretch factor of key paths in T^* of the *RKP-DE* protocol is slightly larger than that of key paths in T_1^* of the *RKP* protocol. We first explain the reason for the decreased stretch factor of key paths in T_1^* by an example. Suppose there exists a key path between a source node and a destination node with multiple logical hops in the *RKP* protocol. Consider a segment (a part of the key path) such that the two end nodes of the segment share keys but are not physical neighbors. In our *RKP-DE* protocol, we can replace the above segment with multiple physical hops by a single logical hop with only two physical hops.³ This can help decrease the number of physical hops and the stretch factor. The reason why the stretch factor of key paths in T_2^* is larger than that of key paths in T_1^* is that with network decoupling, many key paths not identified in the *RKP* protocol can now be constructed by the *RKP-DE* protocol. These key paths in T_2^* tend to have a larger stretch factor.

In the above, we show the derivation of the stretch factor of our *RKP-DE* protocol. Compared with the *RKP-DE* protocol, our *RKP-DEA* protocol tends to increase the stretch factor when increasing the information areas for poor nodes, and it tends to decrease the stretch factor when choosing a subset of key paths for rich nodes. The former is due to the fact that in our adaptive local graphs construction, the extra key paths constructed in larger information areas tend to be longer than those constructed in smaller information areas. The latter is because in our adaptive key paths construction, we tend to choose key paths with fewer logical hops based on (9), which effectively decreases the stretch factor. Overall, the stretch factor of our *RKP-DEA* protocol depends on the protocol parameters discussed in Section 5.

6.3 Algorithm Complexity

There are four components in both our *RKP-DE* protocol and *RKP-DEA* protocol. In the following, we will derive the algorithm complexity of each component.

In the common part of local graphs construction, complexity is decided by local logical graph construction, which is $\Theta(D_p^2 k)$. For the *RKP-DEA* protocol, the execution of deciding the information area size has a complexity upper bounded by $\Theta((MAX_INFOAREA^2 D_p)^2 k)$.

Since the dependency checking in Algorithm 2 is used in the key paths construction in Algorithm 1, we need to obtain the complexity of dependency checking first. The complexity of link dependency checking is $\Theta(h_{avg} k_{avg})$, where h_{avg} is the average number of logical hops on a key path, and k_{avg} is the average number of shared keys on a logical hop. The complexity of path dependency checking is $\Theta(p_{avg}^2 h_{avg}^2 k_{avg})$, where p_{avg} is the average number of logical key paths in a logical key path tree.

After obtaining the complexity of dependency checking above, we can obtain the complexity of the key paths construction in Algorithm 1. The complexity of the logical key path tree construction is $\Theta(p_{avg} h_{avg}^2 k_{avg})$, and that of the

3. For the case where the information area is the one-hop communication range, all nodes on the key path are within the communication range of the source node.

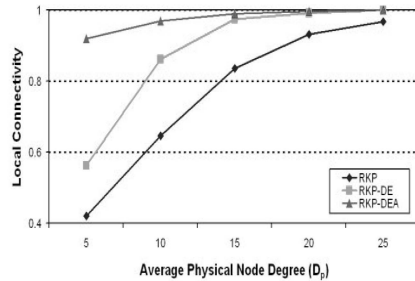


Fig. 6. Sensitivity of local connectivity to the average physical node degree D_p .

physical key paths construction is $\Theta(D_p p_{avg}^2 h_{avg}^2 k_{avg})$ for both protocols.

Our keyshares transmission has a complexity of $\Theta(D_p p_{avg} h_{avg})$ for both protocols. Overall, the complexity of our *RKP-DE* protocol is $\Theta(D_p^2 k + D_p p_{avg}^2 h_{avg}^2 k_{avg})$, and the complexity of our *RKP-DEA* protocol is upper bounded by $\Theta((MAX_INFOAREA^2 D_p)^2 k + D_p p_{avg}^2 h_{avg}^2 k_{avg})$. Based on our analysis and simulation results, the values of p_{avg} , h_{avg} , and k_{avg} are small (less than 10) under reasonable parameters. Therefore, our protocols have acceptable complexity for energy-constrained sensor nodes.

7 PERFORMANCE EVALUATIONS

7.1 Simulation Environment

The sensor network is a square region of size $1,000 m \times 1,000 m$ in which 1,000 sensors are deployed uniformly at random. The communication range r is chosen based on the desired average physical node degree D_p . The following are the default values for the parameters unless otherwise specified: $A = \pi r^2$, $D_p = 10$, $K = 10,000$, $k = 50$, $P_c = 0.99$, and $x = 50$. Each simulation is run 100 times with different random seeds, and the data presented is the average of 100 runs.

7.2 Sensitivity of Connectivity to D_p

7.2.1 Local Connectivity

In Fig. 6, we study the sensitivity of local connectivity to the average physical node degree D_p . We observe that the local connectivity in the *RKP-DE* and *RKP-DEA* protocols is consistently higher than that in the *RKP* protocol. The improvement is in fact more significant for nonhighly dense networks where $D_p < 20$. The relaxation of the constraints in key path construction as a result of network decoupling enables the availability of many more key paths, which greatly enhances the local connectivity. The *RKP-DEA* protocol further enhances local connectivity by increasing the information area for poor nodes.

7.2.2 Global Connectivity

The definition of global connectivity here is the percent of nodes in the largest connected component of the secure network (for example, Fig. 1b or 1c). In Fig. 7, we observe that the global connectivity of the *RKP-DE* and *RKP-DEA* protocols is higher than that of the *RKP* protocol in all situations. The improvement is especially significant in nonhighly dense networks. According to the phase transition

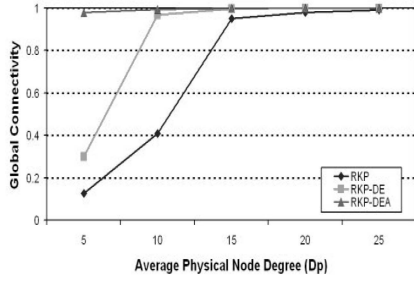


Fig. 7. Sensitivity of global connectivity to the average physical node degree D_p .

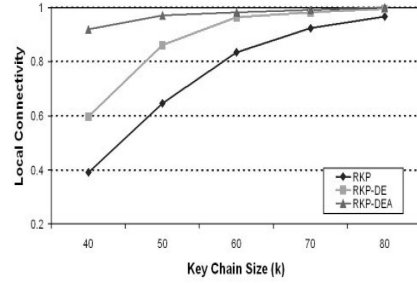


Fig. 9. Sensitivity of local connectivity to the key chain size k .

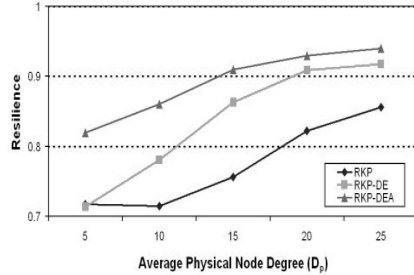


Fig. 8. Sensitivity of resilience to the average physical node degree D_p .

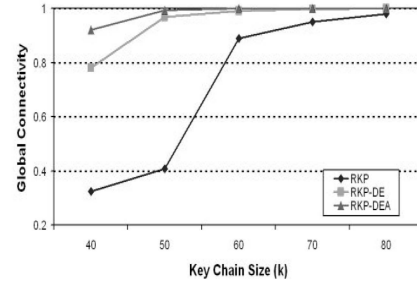


Fig. 10. Sensitivity of global connectivity to the key chain size k .

phenomenon in random graphs [12], 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 occurs when D_p is around 10 compared to the *RKP* protocol when D_p is around 15. With the ability to adjust the information area, such a jump in the *RKP-DEA* protocol occurs when D_p is even smaller than 5.

7.3 Sensitivity of Resilience to D_p

In Fig. 8, we study the sensitivity of resilience to D_p . We see that resilience is higher in *RKP-DE* and *RKP-DEA* compared to that in *RKP* in general. Network decoupling not only increases the number of key paths between two physical neighbor nodes but also decreases the logical hops of many key paths, both of which help enhance the resilience. When the network becomes very sparse, only a single key path can be constructed in most situations in the *RKP-DE* protocol; thus, the improvement diminishes. By adjusting the information area for poor nodes, our *RKP-DEA* protocol can achieve a significant improvement under sparse networks.

7.4 Sensitivity of Connectivity and Resilience to k and x

In Figs. 9, 10, and 11, we study the sensitivity of connectivity and resilience to k and x . In Figs. 9 and 10, we see a similar pattern in the sensitivity of connectivity to k as that to D_p . This is because the increase in k enhances the chance two nodes share keys, which makes the local logical graphs more dense. This can also be achieved by increasing D_p .

In Fig. 11, we study the sensitivity of resilience to k under different values for x . We observe that the resilience of the *RKP-DE* and *RKP-DEA* protocols is better than that of the *RKP* protocol for all cases. When x is relatively small ($x = 50$), the resilience increases with k (for k no more than 250), and the improvement diminishes when k increases. This is because when k increases, the local logical graphs become

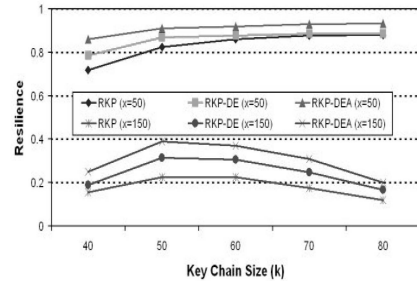


Fig. 11. Sensitivity of resilience to the key chain size k and number of captured nodes x .

more dense, and more key paths can be constructed. Although the increased density improves resilience, it also results in diminishing the performance improvement. On the other hand, when x is relatively large ($x = 150$), the resilience first increases with k and then begins to decrease when k further increases. This is because when both k and x are relatively large, the attacker is able to disclose a significant percent of the predistributed keys, thereby compromising many more established pairwise keys, which degrades the resilience. The threshold value for k , beyond which resilience begins to decrease, will decrease with increase in x . The value of x does not impact connectivity, so we do not show the sensitivity of connectivity to x .

8 RELATED WORK

The *RKP* protocol has received wide acceptance in WSNs and served as a foundation for many other works [5], [7], [8], [9], [10], [14]. In [5] and [9], the performance of the basic *RKP* protocol is enhanced by constructing multiple key paths for pairwise key establishment. With multiple key paths, as long as at least one path is uncompromised, the pairwise key is secure. In [7], [8], [10], and [14], the authors extend the basic *RKP* protocol by predistributing key structures (either polynomials or vectors) instead of keys. When the number of captured nodes is small, this protocol

has much better resilience compared to the basic *RKP* protocol. We point out that in the above works, a very high network density (an average physical node degree between 20 and 250) is assumed to achieve satisfactory performance.

We now discuss two works that share certain similarities with our network decoupling methodology. In Peer Intermediaries for Key Establishment (*PIKE*) in [15], node IDs of all nodes form a two-dimensional virtual ID space with size $\sqrt{N} \times \sqrt{N}$, where N is the number of nodes in the network. Each node is predistributed with a unique pairwise key with each of the nodes whose node IDs lie in the same row or same column of the virtual ID space. For any two neighboring nodes u and v , if their node IDs lie in the same row or same column, they naturally share a pairwise key. Otherwise, there are two other nodes in the network that share predistributed keys with both u and v , each of which can act as a proxy and form a key path with two logical hops between nodes u and v . Another work is the Recursive Key Establishment Protocol (*RKEP*) in [6]. In *RKEP*, a key graph is first constructed based on keysharing among nodes, and then, the key graph is used to establish pairwise keys. Similar to *PIKE*, *RKEP* also guarantees pairwise key establishment between any two nodes.

As we can see, the above two works share some similarities with our network decoupling in that non-neighboring nodes can form a logical hop on a key path if they share predistributed keys. However, there are major differences between these works and our work. Primarily, the core idea of our network decoupling is to decouple the network *after* node deployment, whereas in *PIKE* and *RKEP*, the idea of decoupling is combined with key predistribution *before* node deployment, which not only limits the way to distribute keys but also introduces other limitations, discussed below. Second, *PIKE* and *RKEP* incur much more communication overhead. Both *PIKE* and *RKEP* require networkwise communication for *each* pairwise key establishment, whereas our protocols are localized. Third, both *PIKE* and *RKEP* require a routing structure to be constructed before pairwise key establishment in order to prevent flooding during the search for key paths. However, without the protection of pairwise keys, routing structure construction is prone to attacks. Such a dilemma does not exist in our protocols.

Orthogonal extensions to the basic *RKP* protocol are exploiting certain network properties to enhance performance or decrease overhead. Works like [16], [17], [18], [19], [20], [21], and [22] use power control, node mobility, channel diversity, hierarchy, and deployment knowledge to enhance performance or decrease overhead. We point out that our network decoupling is orthogonal to all the works above and can complement them to achieve further performance improvement and overhead reduction.

9 FINAL REMARKS

In this paper, we proposed *network decoupling* to separate the logical keysharing relationship from the physical neighborhood relationship in random key predistributed sensor networks. We designed two new key management protocols (*RKP-DE* and *RKP-DEA*) in decoupled sensor networks and also designed a set of rules for eliminating dependencies among key paths. We conducted a detailed analysis and extensive simulations to evaluate our proposed solutions from the perspective of connectivity, resilience,

and overhead. Our data showed that a significant performance improvement can be achieved using our solutions in non-highly dense networks with only a mild increase in overhead. Our future work consists of practically implementing our proposed solutions on the existing sensor network testbed at Ohio State University (OSU) [23], [24].

APPENDIX

In this section, we will present the derivation of $P_h(i)$ and $P_h(i|A')$ defined in Section 3. We will first derive some preliminaries, followed by the derivation of $P_h(i)$ and $P_h(i|A')$ using techniques in [11].

A. Preliminaries

Given the key pool size K and the key chain size k ($k \leq K$), the probability that two nodes share t keys is $P_{key}(t) = \binom{K}{t} \cdot \binom{K-t}{2(k-t)} \cdot \binom{2(k-t)}{k-t} / \binom{K}{k}^2$ ($0 \leq t \leq k$).

Each sensor is aware of the keysharing and physical neighborhood information in its *information area* with size A . For ease of exposition, we assume that each sensor is aware of its one-hop information. Therefore, $A = \pi r^2$. Note that our analysis can be applied directly to the situation where multihop information is available. Since n sensors are uniformly deployed at random in the network with area S , the average number of nodes in the information area is D_p , which is given by (ignoring boundary effect) $D_p = \frac{A}{S} \cdot n$. The average number of nodes in the overlapped information areas of two physical neighbor nodes is $D'_p = \frac{A'}{S} \cdot n$, where A' is the average size of the overlapped information areas of two physical neighbor nodes. Since the information area is a circle, we have $A' = \frac{(\pi - \frac{3\sqrt{3}}{4})}{\pi} A = 0.5865A$ (as given in [5]).

B. Derivation of $P_h(i)$ and $P_h(i|A')$

We denote $P_h(i)$ as the probability that a node a can find a logical key path to a physical neighbor node b in our *RKP-DE* protocol within the information area of node a with minimum logical hops i . The expression for $P_h(1)$ is given by

$$P_h(1) = 1 - P_{key}(0) = 1 - \binom{K}{2k} \cdot \binom{2k}{k} / \binom{K}{k}^2. \quad (16)$$

In order to analyze $P_h(i)$ ($i > 1$), we divide the nodes in the information area of node a (except nodes a and b) into one of the groups $G(a, j)$ ($j \geq 1$). A node s is in group $G(a, j)$ if node a can find a logical key path from itself to node s within its information area with j logical hops, where j is minimum.

We first analyze $P_h(2)$. The probability that there are n_1 ($1 \leq n_1 \leq D_p - 1$) nodes in $G(a, 1)$, given that node b does not share a key with node a , is $\binom{D_p-1}{n_1} (P_h(1))^{n_1} (1 - P_h(1))^{D_p-1-n_1}$. The probability that at least one of these n_1 nodes shares keys with node b is $1 - (1 - P_h(1))^{n_1}$. Hence, $P_h(2)$ is

4. Node b is not in $G(a, 1)$. Therefore, n_1 can be $D_p - 1$ at most.

$$P_h(2) = (1 - P_h(1)) \cdot \sum_{n_1=1}^{D_p-1} \left(\binom{D_p-1}{n_1} (P_h(1))^{n_1} \right. \\ \left. (1 - P_h(1))^{D_p-1-n_1} \cdot (1 - (1 - P_h(1))^{n_1}) \right). \quad (17)$$

We now analyze $P_h(3)$. The probability that there are n_1 ($1 \leq n_1 \leq D_p - 2$)⁵ nodes in $G(a, 1)$, given that there is no key path between nodes a and b within the communication range of node a with fewer than three logical hops, is $\binom{D_p-1}{n_1} v^{n_1} (1 - P_h(1))^{D_p-1-n_1}$. The expression here is different from that in deriving $P_h(2)$ because in this case, nodes in $G(a, 1)$ do not share a key with node b . Otherwise, a logical key path with fewer than three logical hops exists. Denoting v as the probability that a node shares keys with node a but does not share a key with node b , given that nodes a and b do not share a key, we have $v = \left(\binom{K-k}{k} - \binom{K-2k}{k} \right) / \binom{K}{k}$. Denote w as the probability that there is at least one node in $G(a, 2)$, given that there are n_1 nodes in $G(a, 1)$, and at least one of the nodes in $G(a, 2)$ shares keys with node b . Thus, the minimum number of logical hops of the key path between nodes a and b is three. Then, w is given by

$$w = \sum_{n_2=1}^{D_p-1-n_1} \left(\binom{D_p-1-n_1}{n_2} (1 - (1 - P_h(1))^{n_1})^{n_2} \right. \\ \left. \left((1 - P_h(1))^{n_1} \right)^{D_p-1-n_1-n_2} \cdot (1 - (1 - P_h(1))^{n_2}) \right). \quad (18)$$

Finally, the expression for $P_h(3)$ is given by

$$P_h(3) = (1 - P_h(1)) \cdot \sum_{n_1=1}^{D_p-2} \left(\binom{D_p-1}{n_1} v^{n_1} (1 - P_h(1))^{D_p-1-n_1} \cdot w \right). \quad (19)$$

The derivation of $P_h(l)$ ($l > 3$) is similar to $P_h(3)$. We denote $y(i)$ ($2 \leq i \leq l - 2$) as the probability that there is at least one node in $G(a, i)$ given n_j ($1 \leq j \leq i - 1$) nodes in $G(a, j)$ ⁶ and denote z as the probability that at least one node is in $G(a, l - 1)$ and shares keys with node b . We then have

$$y(i) = \sum_{n_i=1}^{D_p-1-\sum_{j=1}^{i-1} n_j - (l-1-i)} \left(\binom{D_p-1-\sum_{j=1}^{i-1} n_j}{n_i} \right. \\ \left((1 - (1 - P_h(1))^{n_{i-1}}) (1 - P_h(1)) \right)^{n_i} \\ \cdot \left((1 - P_h(1))^{n_{i-1}} \right)^{D_p-1-\sum_{j=1}^i n_j} \Big), \quad (20)$$

5. Node b is not in $G(a, 1)$, and at least one other node is in $G(a, 2)$ (thus, not in $G(a, 1)$). Therefore, n_1 can be $D_p - 2$ at most.

6. None of the nodes in $G(a, i)$ ($1 \leq i \leq l - 2$) shares keys with node b ; otherwise, a key path with less than l logical hops exists.

$$z = \sum_{n_{l-1}=1}^{D_p-1-\sum_{j=1}^{l-2} n_j} \left(\binom{D_p-1-\sum_{j=1}^{l-2} n_j}{n_{l-1}} (1 - (1 - P_h(1))^{n_{l-2}})^{n_{l-1}} \right. \\ \left((1 - P_h(1))^{n_{l-2}} \right)^{D_p-1-\sum_{j=1}^{l-1} n_j} \\ \cdot (1 - (1 - P_h(1))^{n_{l-1}}) \Big). \quad (21)$$

The general form of $P_h(l)$ ($l > 3$) is given by

$$P_h(l) = (1 - P_h(1)) \cdot \sum_{n_1=1}^{D_p-1-(l-2)} \left(\binom{D_p-1}{n_1} v^{n_1} \right. \\ \left. (1 - P_h(1))^{D_p-1-n_1} \cdot \prod_{i=2}^{l-2} y(i) \cdot z \right). \quad (22)$$

The expressions of $P_h(i|A')$ are the same as those of $P_h(i)$, except replace D_p with D'_p .

REFERENCES

- [1] W. Diffie and M.E. Hellman, "New Directions in Cryptography," *IEEE Trans. Information Theory*, vol. 22, no. 6, pp. 644-654, Nov. 1976.
- [2] R.L. Rivest, A. Shamir, and L.M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM*, vol. 21, no. 2, pp. 120-126, Feb. 1978.
- [3] B.C. Neuman and T. Tso, "Kerberos: An Authentication Service for Computer Networks," *IEEE Comm. Magazine*, vol. 32, no. 9, pp. 33-38, Sept. 1994.
- [4] L. Eschenauer and V.D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," *Proc. Ninth ACM Conf. Computer and Comm. Security (CCS '02)*, Nov. 2002.
- [5] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *Proc. IEEE Symp. 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," *Proc. Third ACM Conf. Embedded Networked Sensor Systems (SenSys '05)*, Nov. 2005.
- [7] W. Du, J. Deng, Y.S. Han, and P.K. Varshney, "A Pairwise Key Predistribution Scheme for Wireless Sensor Networks," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03)*, Oct. 2003.
- [8] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03)*, Oct. 2003.
- [9] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishing Pairwise Keys for Secure Communication in Ad Hoc Networks: A Probabilistic Approach," *Proc. 11th IEEE Int'l Conf. Network Protocols (ICNP '03)*, Nov. 2003.
- [10] F. Delgosha and F. Fekri, "Threshold Key-Establishment in Distributed Sensor Networks Using a Multivariate Scheme," *Proc. INFOCOM '06*, Apr. 2006.
- [11] D. Huang, M. Mehta, D. Medhi, and L. Harn, "Modeling Pairwise Key Establishment for Random Key Predistribution in Large-Scale Sensor Networks," technical report, Dept. of Computer Science and Electrical Eng., Univ. of Missouri, Kansas City, <http://conrel.sice.umkc.edu/HRP/modellingpair-wisekeyestablishmentschemes-v2.pdf>, Mar. 2005.
- [12] J. Spencer, *The Strange Logic of Random Graphs*, Algorithms and Combinatorics 22. Springer, 2000.
- [13] W. Gu, X. Bai, S. Chellappan, and D. Xuan, "Network Decoupling for Secure Communications in Wireless Sensor Networks," *Proc. 14th IEEE Int'l Workshop Quality of Service (IWQoS '06)*, June 2006.
- [14] F. Delgosha and F. Fekri, "Key Predistribution in Wireless Sensor Networks Using Multivariate Polynomials," *Proc. Second IEEE Comm. Soc. Conf. Sensor and Ad Hoc Comm. and Networks (SECON '05)*, Sept. 2005.
- [15] H. Chan and A. Perrig, "PIKE: Peer Intermediaries for Key Establishment," *Proc. INFOCOM '05*, Mar. 2005.

- [16] J. Hwang and Y. Kim, "Revisiting Random Key Pre-Distribution Schemes for Wireless Sensor Networks," *Proc. Second ACM Workshop Security of Ad Hoc and Sensor Networks (SASN '04)*, Oct. 2004.
- [17] Y. Mao and M. Wu, "Coordinated Sensor Deployment for Improving Secure Communications and Sensing Coverage," *Proc. Third ACM Workshop Security of Ad Hoc and Sensor Networks (SASN '05)*, Nov. 2005.
- [18] M. Miller and N. Vaidya, "Leveraging Channel Diversity for Key Establishment in Wireless Sensor Networks," *Proc. INFOCOM '06*, Apr. 2006.
- [19] P. Traynor, H. Choi, G. Cao, S. Zhu, and T.L. Porta, "Establishing Pair-Wise Keys in Heterogeneous Sensor Networks," *Proc. INFOCOM '06*, Apr. 2006.
- [20] D. Huang, M. Mehta, D. Medhi, and L. Harn, "Location-Aware Key Management Scheme for Wireless Sensor Networks," *Proc. Second ACM Workshop Security of Ad Hoc and Sensor Networks (SASN '04)*, Oct. 2004.
- [21] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney, "A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge," *Proc. INFOCOM '04*, Mar. 2004.
- [22] D. Liu, P. Ning, and W. Du, "Group-Based Key Pre-Distribution in Wireless Sensor Networks," *Proc. ACM Workshop Wireless Security (WiSe '05)*, Sept. 2005.
- [23] E. Ertin, A. Arora, R. Ramnath, and M. Nesterenko, "Kansei: A Testbed for Sensing at Scale," *Proc. Fourth Symp. Information Processing in Sensor Networks (IPSN '06) Special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS)*, Apr. 2006.
- [24] A. Arora, E. Ertin, R. Ramnath, M. Nesterenko, and W. Leal, "Kansei: A High-Fidelity Sensing Testbed," *IEEE Internet Computing*, special issue on large-scale sensor networks, vol. 10, no. 2, pp. 35-47, Mar./Apr. 2006.



Sriram Chellappan received the bachelor's degree in instrumentation and control engineering from the University of Madras and the master's degree in electrical engineering from Ohio-State University. He is a PhD candidate in the Department of Computer Science and Engineering at Ohio State University. His current research interests are in network security, distributed systems and wireless networks. He is a student member of the IEEE.

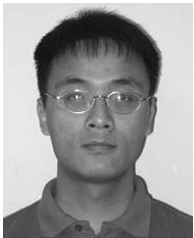


Dong Xuan received the BS and MS degrees in electronic engineering from Shanghai Jiao Tong University (SJTU), China, in 1990 and 1993, respectively, and the PhD degree in computer engineering from Texas A&M University in 2001. Currently, he is an assistant professor in the Department of Computer Science and Engineering, Ohio State University. He was on the Faculty of Electronic Engineering at SJTU from 1993 to 1997. In 1997, he worked as a visiting research scholar in the Department of Computer Science, City University of Hong Kong. From 1998 to 2001, he was a research assistant/associate in the Real-Time Systems Group of the Department of Computer Science, Texas A&M University. He is a recipient of the US National Science Foundation (NSF) Faculty Early Career Development (CAREER) Award. His research interests include real-time computing and communications, network security, sensor networks, and distributed systems. He is a member of the IEEE.

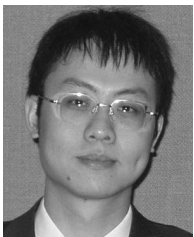


Weijia Jia received the BSc and MSc degrees from Center South University, China, and the PhD degree from the Polytechnic Faculty of Mons, Belgium, all in computer science. He joined the German National Research Center for Information Science (GMD), St. Augustin, in 1993 as a research fellow. Since 1995, he joined the Department of Computer Science, City University of Hong Kong, as an associate professor. His research interests include wireless communication and networks, distributed systems, multicast, and anycast quality-of-service (QoS) routing protocols for the Internet. In these fields, he has published more than 200 papers and books/chapters in international journals and conference proceedings. He has been the principal investigator of more than 20 research projects supported by Research Grant Council (RGC) Research Grants, Hong Kong and Strategic Research Grants, City University of Hong Kong. Currently, he is in charge of a HK \$10 million ITF project supported by the Hong Kong Government for the development of next generation ubiquitous communication platforms. He has served as a program committee (PC) cochair and a PC member for various IEEE international conferences. He is a member of the IEEE.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.



Wenjun Gu received the BS and MS degrees in electronic engineering from Shanghai Jiao Tong University (SJTU), China, in 2000 and 2003, respectively. He is a PhD student in the Department of Computer Science and Engineering at Ohio State University. His current research interests are in wireless networks, network security and distributed systems.



Xiaole Bai received the BS degree in optical fiber communication from the Electrical Engineering Department, Southeast University, China, in 1999 and the MS degree in communication and networking from the Helsinki University of Technology, Finland, in 2003. From September 2003 to September 2004, he was working as a research scientist in the Networking Laboratory at the Helsinki University of Technology. He is now working toward the PhD degree in Computer Science and Engineering Department at Ohio State University. His research interests include distributed computing, network architecture, and distributed algorithms.