A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge

Wenliang Du*, Jing Deng*, Yunghsiang S. Han†, Shigang Chen‡, and Pramod K. Varshney*
*Department of Electrical Engineering and Computer Science
Syracuse University, Syracuse, NY 13244-1240, USA
Email: {wedu, jdeng01, varshney}@ecs.syr.edu
†Department of Computer Science and Information Engineering
National Chi Nan University, Taiwan, R.O.C.
Email: yshan@csie.ncnu.edu.tw
‡Department of Computer and Information Science and Engineering
University of Florida, Gainesville, FL 32611-6120, USA
Email: sgchen@cise.ufl.edu

***Abstract*— To achieve security in wireless sensor networks, it is important to be able to encrypt messages sent among sensor nodes. Keys for encryption purposes must be agreed upon by communicating nodes. Due to resource constraints, achieving such key agreement in wireless sensor networks is non-trivial. Many key agreement schemes used in general networks, such as Diffie-Hellman and public-key based schemes, are not suitable for wireless sensor networks. Pre-distribution of secret keys for all pairs of nodes is not viable due to the large amount of memory used when the network size is large. Recently, a random key pre-distribution scheme and its improvements have been proposed.**

A common assumption made by these random key pre-distribution schemes is that no deployment knowledge is available. Noticing that in many practical scenarios, certain deployment knowledge may be available *a priori*, we propose a novel random key pre-distribution scheme that exploits deployment knowledge and avoids unnecessary key assignments. We show that the performance (including connectivity, memory usage, and network resilience against node capture) of sensor networks can be substantially improved with the use of our proposed scheme. The scheme and its detailed performance evaluation are presented in this paper.

I. Introduction

Recent advances in electronic and computer technologies have paved the way for the proliferation of wireless sensor networks (WSN). Sensor networks usually consist of a large number of ultra-small autonomous devices. Each device, called a sensor node, is battery powered and equipped with integrated sensors, data processing capabilities, and short-range radio communications. In typical application scenarios, sensor nodes are spread randomly over the deployment region under scrutiny and collect sensor data. Examples of sensor network projects include SmartDust [1] and WINS [2].

Sensor networks are being deployed for a wide variety of applications [3], including military sensing and tracking, environment monitoring, patient monitoring and tracking, smart environments, etc. When sensor networks are deployed in a hostile environment, security becomes extremely important, as they are prone to different types of malicious attacks. For example, an adversary can easily listen to the traffic, impersonate one of the network nodes[1], or intentionally provide misleading information to other nodes. To provide security, communication should be encrypted and authenticated. An open research problem is how to bootstrap secure communications among sensor nodes, i.e. how to set up secret keys among communicating nodes?

This key agreement problem is a part of the *key management* problem, which has been widely studied in general network environments. There are three types of general key agreement schemes: trusted-server scheme, self-enforcing scheme, and key pre-distribution scheme. The *trusted-server* scheme depends on a trusted server for key agreement between nodes, e.g., Kerberos [4]. This type of scheme is not suitable for sensor networks because there is usually no trusted infrastructure in sensor networks. The *self-enforcing* scheme depends on asymmetric cryptography, such as key agreement using public key certificates. However, limited computation and energy resources of sensor nodes often make it undesirable to use public key algorithms, such as Diffie-Hellman key agreement [5] or RSA [6], as pointed out in [7]. The third type of key agreement scheme is key *pre-distribution*, where key information is distributed among all sensor nodes prior to deployment. If we know which nodes are more likely to stay in the same neighborhood before deployment, keys can be decided *a priori*. However, because of the randomness of the deployment, knowing the set of neighbors deterministically might not be feasible.

There exist a number of key pre-distribution schemes. A naive solution is to let all the nodes carry a *master* secret key. Any pair of nodes can use this global master secret key to achieve key agreement and obtain a new pairwise key. This scheme does not exhibit desirable network resilience: if one node is compromised, the security of the entire sensor network will be compromised. Some existing studies suggest storing the master key in tamper-resistant hardware to reduce

[1] In this paper, we use the terms sensors, sensor nodes, and nodes interchangeably.

0-7803-8355-9/04/$20.00 ©2004 IEEE.

the risk, but this increases the cost and energy consumption of each sensor. Furthermore, tamper-resistant hardware might not always be safe [8]. Another key pre-distribution scheme is to let each sensor carry $N-1$ secret pairwise keys, each of which is known only to this sensor and one of the other $N-1$ sensors (assuming N is the total number of sensors). The resilience of this scheme is perfect because compromising one node does not affect the security of communications among other nodes; however, this scheme is impractical for sensors with an extremely limited amount of memory because N could be large. Moreover, adding new nodes to a pre-existing sensor network is difficult because the existing nodes do not have the new nodes' keys.

Eschenauer and Gligor recently proposed a random key pre-distribution scheme: before deployment, each sensor node receives a random subset of keys from a large key pool. To agree on a key for communication, two nodes find one common key within their subsets and use that key as their shared secret key [9]. Our scheme is based on the Eschenauer-Gligor scheme, and we refer to this scheme as the *basic scheme* throughout this paper. An overview of it is given in Section II. The Eschenauer-Gligor scheme is further improved by Chan, Perrig, and Song [10], by Du, Deng, Han, and Varshney [11], and by Liu and Ning [12].

A. Outline of Our Scheme

Although the proposed schemes provided viable solutions to the key pre-distribution problem, they have not exploited a piece of information that might significantly improve their performance. This piece of information is *node deployment knowledge*, which, in practice, can be derived from the way that nodes are deployed.

Let us look at a deployment method that uses an airplane to deploy sensor nodes. The sensors are first pre-arranged in a sequence of smaller groups. These groups are dropped out of the airplane sequentially as the plane flies forward. This is analogous to parachuting troops or dropping cargo in a sequence. The sensor groups that are dropped next to each other have a better chance to be close to each other on the ground. This spatial relation between sensors derived prior to deployment can be useful for key pre-distribution. The goal of this paper is to show that knowledge regarding the actual non-uniform sensor deployment can help us improve the performance of a key pre-distribution scheme.

Knowing which sensors are close to each other is important to key pre-distribution. In sensor networks, long distance peer-to-peer secure communication between sensor nodes is rare and unnecessary in many applications. The primary goal of secure communication in wireless sensor networks is to provide such communications among neighboring nodes. Therefore, the most important knowledge that can benefit a key-predistribution scheme is the knowledge about *the nodes that are likely to be the neighbors of each sensor node*. When we know deterministically the neighbors of each node in the network, key pre-distribution becomes trivial: for each node n, we just need to generate a pairwise key between n and each of its neighboring nodes, and save these keys in n's memory. This guarantees that each node can establish a secure channel with each of its neighbors after deployment.

However, because of the randomness of deployment, it is unrealistic to know the exact set of neighbors of each node, but knowing the set of *possible* or likely neighbors for each node is much more realistic. However, the number of possible neighbors can be very large and it may not be feasible for a sensor to store the secret keys for each potential neighbor due to memory limitations. This problem can be solved using the random key pre-distribution scheme [9], i.e., instead of guaranteeing that any two neighboring nodes can find a common secret key with 100% certainty, we only guarantee that any two neighboring nodes can find a common secret key with a certain probability p. In this paper, we exploit deployment knowledge in the random key pre-distribution scheme [9], such that the probability p can be maximized while the other performance metrics (such as security and memory usage) are not degraded.

Deployment knowledge can be modeled using probability density functions (pdfs). When the pdf is uniform, no information can be gained on where a node is more likely to reside. All the existing key pre-distribution schemes assume such a uniform distribution. In this paper, we look at non-uniform pdf functions. Since the distribution is different from uniform distribution, it is equivalent to saying that we know that a sensor is more likely to be deployed in certain areas. We will show how this knowledge can help improve the random key pre-distribution scheme proposed by Eschenauer and Gligor in [9]. To demonstrate the effectiveness of our method, we studied a specific distribution, the Normal (Gaussian) distribution, in great depth. Our results show substantial improvement over existing schemes.

B. Main Contributions of Our Scheme

The main contributions of this paper are summarized in the following:

1) We model node deployment knowledge in a wireless sensor network, and develop a key pre-distribution scheme based on this model. This is the first attempt at the use of deployment knowledge while developing a key pre-distribution scheme.
2) We show that key pre-distribution with deployment knowledge can substantially improve a network's connectivity (in terms of secure links) and resilience against node capture, and reduce the amount of memory required.

C. Related Work

The Eschenauer-Gligor scheme [9] have been described earlier in this section. We will give a more detailed description of this scheme in Section II. Based on the Eschenauer-Gligor scheme, Chan, Perrig, and Song proposed a q-composite random key pre-distribution scheme [10]. The difference between this scheme and the Eschenauer-Gligor scheme is that q common keys ($q \geq 1$), instead of just a single one, are needed to establish secure communications between a pair of nodes. It

is shown that, by increasing the value of q, network resilience against node capture is improved, i.e., an attacker has to compromise many more nodes to achieve a high probability of compromised communication.

Du, Deng, Han, and Varshney proposed a new key pre-distribution scheme [11], which substantially improves the resilience of the network compared to the existing schemes. This scheme exhibits a nice threshold property: when the number of compromised nodes is less than the threshold, the probability that any nodes other than these compromised nodes are affected is close to zero. This desirable property lowers the initial payoff of smaller scale network breaches to an adversary, and makes it necessary for the adversary to attack a significant proportion of the network. A similar method is also developed by Liu and Ning [12].

The ideas described in this paper can be applied to all of the above pre-distribution schemes to further improve their performance.

Blundo et al. proposed several schemes which allow any group of t parties to compute a common key while being secure against collusion between some of them [13]. These schemes focus on saving communication costs while memory constraints are not placed on group members.

Perrig et al. proposed SPINS, a security architecture specifically designed for sensor networks [7]. In SPINS, each sensor node shares a secret key with the base station. Two sensor nodes cannot directly establish a secret key. However, they can use the base station as a trusted third party to set up the secret key.

Several other key distribution schemes have been proposed for mobile computing, although they are not specifically targeted at sensor networks. Tatebayashi, Matsuzaki, and Newman consider key distribution for resource-starved devices in a mobile environment [14]. This work is further improved by Park et al. [15]. Other key agreement and authentication protocols include the one by Beller and Yacobi [16]. A survey on key distribution and authentication for resource-starved devices in mobile environments is given in [17]. The majority of these approaches rely on asymmetric cryptography, which is not a feasible solution for sensor networks [7]. Several other methods based on asymmetric cryptography are also proposed: Zhou and Hass propose a secure ad hoc network using secret sharing and threshold cryptography [18]. Kong et al. also propose localized public-key infrastructure mechanisms, based on secret sharing schemes [19].

Stajanor and Anderson studied the issues of bootstrapping security devices, and they proposed a solution that requires physical contact of the new device with a master device to imprint the trusted and secret information [20]. Key pre-distribution is similar to the "imprinting" process, but their focuses are different.

II. The Eschenauer-Gligor random key pre-distribution scheme

The *basic scheme* proposed in [9] consists of three phases: key pre-distribution, shared-key discovery, and path-key establishment.

In the *key pre-distribution phase*, each sensor node randomly selects m distinct cryptographic keys from a key pool S, and stores them in its memory. This set of m keys is called the node's *key ring*. The number of keys in the key pool, $|S|$, is chosen such that two random subsets of size m in S share at least one key with some probability p.

After the nodes are deployed, a *key-setup phase* is performed. During this phase, each pair of neighboring nodes attempt to find a common key that they share. If such a key exists, the key is used to secure the communication link between these two nodes. After key-setup is complete, a connected graph of secure links is formed. Nodes can then set up *path keys* with their neighbors with whom they do not share keys. If the graph is connected, a path can always be found from a source node to any of its neighbors. The source node can then generate a path key and send it securely via the path to the target node.

The size of the key pool S is critical to both the connectivity and the resilience of the scheme. *Connectivity* is defined as the probability that any two neighboring nodes share one key. *Resilience* is defined as the fraction of the secure links that are compromised after a certain number of nodes are captured by the adversaries.

At one extreme, if the size of S is one, i.e., $|S| = 1$, the scheme is actually reduced to the naive master-key scheme. This scheme yields a high connectivity, but it is not resilient against node capture because the capture of one node can compromise the whole network. At the other extreme, if the key pool is very large, e.g. $|S| = 100,000$, resilience becomes much better, but connectivity of the sensor network becomes low. For example, as indicated in [9], in this case, even when each sensor selects $m = 200$ keys from this large key pool S, the probability that any two neighboring nodes share at least one key is only 0.33.

How can we pick a large key pool while still maintaining high connectivity? In this paper, we use deployment knowledge to solve this problem.

III. Modeling of the Deployment Knowledge

We assume that sensor nodes are static once they are deployed. We define *deployment point* as the point location where a sensor is to be deployed. This is not the location where this sensor finally resides. The sensor node can reside at points around this point according to a certain pdf and this point is the mean of the pdf. As an example, let us consider the case where sensors are deployed by dropping them from a helicopter. The deployment point is the location of the helicopter. We also define *resident point* as the point location where a sensor finally resides.

A. A General Deployment Model

Assume that the target deployment area is a two-dimensional rectangular region with size $X \times Y$ and the origin point is the upper left corner. The pdf for the location of node i, for $i = 1, \ldots, N$, over the two-dimensional region

is given by $f_i(x, y)$, where $x \in [0, X]$ and $y \in [0, Y]$. With this general model, the existing key pre-distribution schemes for sensor networks are special cases: they all assume that $f_i(x, y) = \frac{1}{XY}$ for $x \in [0, X]$, $y \in [0, Y]$ and $1 \leq i \leq N$, i.e., all sensor nodes are uniformly distributed over the whole deployment region.

B. Group-based Deployment Model

The above problem defines a general deployment model, in which nodes are deployed individually, thus they may have different pdfs. In practice, it is quite common that nodes are deployed in groups, i.e., a group of sensors are deployed at a single deployment point, and the pdfs of the final resident points of all the sensors in each batch (or group) are the same.

In this work, we assume such a group-based deployment, and we model the deployment knowledge in the following (we call this model the *group-based deployment model*):

1) N sensor nodes to be deployed are divided into $t \times n$ equal size groups so that each group, $G_{i,j}$, for $i = 1, \ldots, t$ and $j = 1, \ldots, n$, is deployed from the deployment point with index (i, j). Let (x_i, y_j) represent the deployment point for group $G_{i,j}$.
2) The deployment points are arranged in a grid. Note that the scheme we developed for grid-based deployment can be easily extended to different deployment strategies. We choose this specific strategy because it is quite common in realistic scenarios.
3) During deployment, the resident points of the node k in group $G_{i,j}$ follow the pdf $f_k^{ij}(x, y|k \in G_{i,j}) = f(x - x_i, y - y_j)$. An example of the pdf $f(x, y)$ is a two-dimensional Gaussian distribution.

When $f(x, y)$ is a uniform distribution over the deployment region, we do not know which nodes are more likely to be close to each other *a priori* because the resident point of a node can be anywhere within the region with the same probability. However, when $f(x, y)$ is a non-uniform distribution, we can determine which nodes are more likely to be close to each other. For example, with Gaussian distribution, we know that the distance between a resident point and the deployment point is less than 3σ with probability 0.9987 (where σ is the standard deviation of the Gaussian distribution). If the deployment points of two groups are 6σ away, then the probability for two nodes from these two different groups to be located near each other is very low. Therefore, the probability that two nodes from two different groups become neighbors decreases with an increase of the distance between the two deployment points.

Recall that in the basic random key pre-distribution scheme [9], when the size of the key pool S becomes smaller, connectivity increases. Since the basic scheme assumes no deployment knowledge (i.e. the distribution $f(x, y)$ is uniform), every node should choose from the same key pool because they are equally likely to be neighbors. However, as we have discussed, when the function $f(x, y)$ is non-uniform, we know that nodes from a specific group are more likely to be neighbors of nodes from the same group and those from nearby groups. Therefore, when two groups are far away from each other, their key pools could be different, rather than the same global key pool S.

We use $S_{i,j}$ to represent the key pool used by group $G_{i,j}$; the union of $S_{i,j}$ (for $i = 1, \ldots, t$ and $j = 1, \ldots, n$) equals S. We use $|S_c|$ to represent the size of $S_{i,j}$ (we select all $S_{i,j}$'s with the same size in this paper). Based on a specific deployment distribution, we can develop a scheme, such that when the deployment points of two groups $G_{a,b}$ and $G_{c,d}$ are farther away from each other, the amount of overlap between $S_{a,b}$ and $S_{c,d}$ becomes smaller or zero.

C. Deployment Distribution

There are many different ways to deploy sensor networks, for example, sensors could be deployed using an airborne vehicle. The actual model for deployment distribution depends on the deployment method.

In this paper, we model the sensor deployment distribution as a Gaussian distribution (also called Normal distribution). Gaussian distribution is widely studied and used in practice. Although we only employ the Gaussian distribution in this paper, our methodology can also be applied to other distributions.

We assume that the deployment distribution for any node k in group $G_{i,j}$ follows a two-dimensional Gaussian distribution. When the deployment point of group $G_{i,j}$ is at (x_i, y_j), we have $\mu = (x_i, y_j)$ and the pdf for node k in group $G_{i,j}$ is the following [21]:

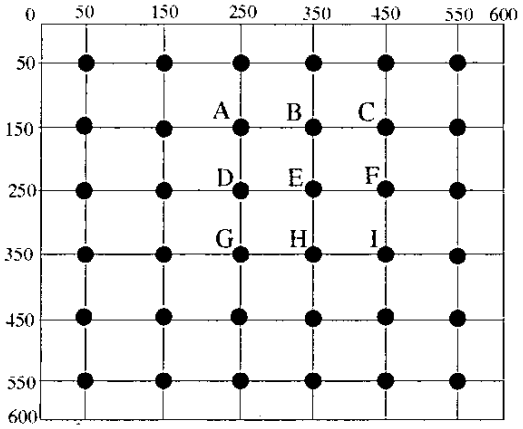
$$\begin{aligned} f_k^{ij}(x, y|k \in G_{i,j}) &= \frac{1}{2\pi\sigma^2} e^{-[(x-x_i)^2+(y-y_j)^2]/2\sigma^2} \\ &= f(x - x_i, y - y_j), \end{aligned}$$

where $f(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$. Without loss of generality, we assume that the pdf for each group is identical, so we use $f_k(x, y|k \in G_{ij})$ instead of $f_k^{ij}(x, y|k \in G_{i,j})$ throughout this paper.

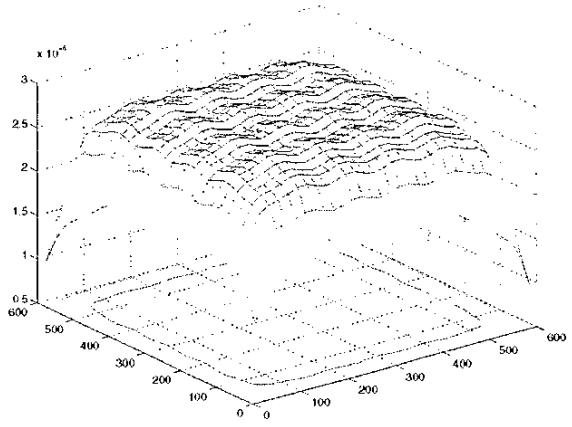
Although the distribution function for each single group is not uniform, we still want the sensor nodes to be evenly deployed throughout the entire region. By choosing a proper distance between the neighboring deployment points with respect to the value of σ in the pdf of each deployment group, the probability of finding a node in each small region can be made approximately equal. Assuming that a sensor node is selected to be in a given group with an equal probability, $\frac{1}{tn}$, the average deployment distribution (pdf) of any sensor node over the entire region is:

$$f_{overall}(x, y) = \sum_{i=1}^{t} \sum_{j=1}^{n} \frac{1}{t \cdot n} \cdot f_k(x, y|k \in G_{i,j}). \quad (1)$$

To see the overall distribution of sensor nodes over the entire deployment region, we have plotted $f_{overall}$ in Eq. (1) for $6 \times 6 = 36$ groups over a $600m \times 600m$ square region with the deployment points $2\sigma = 100m$ apart (assuming $\sigma = 50$). Fig. 1(a) shows all the deployment points, and Fig. 1(b) shows the overall pdf. From Fig. 1(b), we can see that the pdf is



(a) Deployment points (each dot represents a deployment point).



(b) Deployment distribution on the entire region using the deployment strategy modeled by (a).

Fig. 1. Node Deployment

almost flat (i.e. nodes are fairly evenly distributed) in the whole region except near the boundaries.

IV. Key Pre-distribution using Deployment Knowledge

Based on the deployment model described in the previous section, we propose a new random key pre-distribution scheme, which takes advantage of deployment knowledge. We assume that the sensor nodes are evenly divided into $t \times n$ groups $G_{i,j}$, for $i = 1, \ldots, t$, and $j = 1, \ldots, n$. Assume that the global key pool is S with size $|S|$, and also assume that the deployment points are arranged in a grid depicted in Fig. 1(a). Each node carries m keys.

A. Key Pre-distribution Scheme

The goal of this scheme is to allow sensor nodes to find a common secret key with each of their neighbors after deployment. Our scheme consists of three phases: key pre-distribution, shared-key discovery, and path-key establishment. The last two phases are exactly the same as the basic scheme [9], but because of deployment knowledge, the first phase is considerably different from the basic scheme.

Step 1: Key Pre-distribution phase. This phase is conducted offline and before the sensors are deployed. First we need to divide the key pool S into $t \times n$ key pools $S_{i,j}$ (for $i = 1, \ldots, t$ and $j = 1, \ldots, n$), with $S_{i,j}$ corresponding to the deployment group $G_{i,j}$. We say that two key pools are neighbors (or near each other) if their corresponding deployment groups are deployed in neighboring (or nearby) locations. The goal of setting up the key pools $S_{i,j}$ is to allow the nearby key pools to share more keys, while pools far away from each other share fewer keys or no keys at all. The key-pool setup step will be discussed in detail later.

After the key pools are set up, for each sensor node in the deployment group $G_{i,j}$, we randomly select m keys from its corresponding key pool $S_{i,j}$, and load those keys into the memory of the node.

Step 2: Shared-key discovery phase. After deployment, each node needs to discover whether it shares any keys with its neighbors. To do this, each node broadcasts a message containing the indices of the keys it carries. Each neighboring node can use these broadcast messages to find out if there exists a common key it shares with the broadcasting node. If such a key exists, the neighboring node uses this key to secure its communication channel with the broadcasting node. If we are concerned about disclosing the indices of the keys each node carries, we can use the challenge-response technique to avoid sending the indices [9], namely for every key K_i on a key ring, each node could broadcast a list α, $E_{K_i}(\alpha)$, $i = 1, \ldots, k$, where α is a challenge. The decryption of $E_{K_i}(\alpha)$ with the proper key by a recipient would reveal the challenge α and establish a shared key with the broadcasting node.

After the above step, the entire sensor network forms a *Key-Sharing graph* G, which is defined in the following:

> *Definition 1:* (Key-Sharing Graph) Let V represent all the nodes in the sensor network. A Key-Sharing graph $G(V, E)$ is constructed in the following manner: For any two nodes i and j in V, there exists an edge between them if and only if (1) nodes i and j have at least one common key, and (2) nodes i and j can reach each other within the wireless transmission range, i.e., in a single hop.

Step 3: Path-key establishment phase. It is possible that two neighboring nodes cannot find any common keys between them. In this case, they need to find a secure way to agree upon a common key. We now show how two neighboring nodes, i and j, who do not share a common key could still come up with a secret key between them. The idea is to use the secure

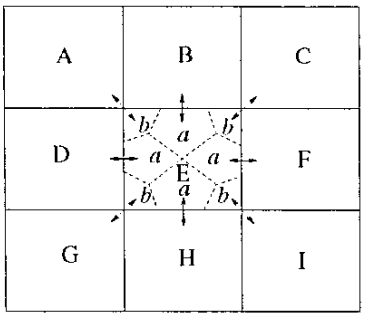


Fig. 2. Shared keys between neighboring key pools

channels that have already been established in the key-sharing graph G: as long as the graph is connected, two neighboring nodes i and j can always find a path in G from i to j. Assume that the path is $i, v_1, \ldots, v_t, j$. To find a common secret key between i and j, i first generates a random key K. Then i sends the key to v_1 using the secure link between i and v_1; v_1 forwards the key to v_2 using the secure link between v_1 and v_2, and so on until j receives the key from v_t. Nodes i and j use this secret key K as their pairwise key. Because the key is always forwarded over a secure link, no nodes beyond this path can find out the key.

To find such a secure path for nodes i and j, the easiest way is to use flooding [22], a common technique used in multihop wireless networks. As we will show later in our analysis, in practice, the probability that the secure path between i and j is within three hops is very high (close to one). Therefore, we can always limit the lifetime of the flooding message to three hops to reduce flooding overhead.

B. Setting Up Key Pools

Next, we show how to assign keys to each key pool $S_{i,j}$, for $i = 1, \ldots, t$ and $j = 1, \ldots, n$, such that key pools corresponding to nearby deployment points have a certain number of common keys. In our scheme, we have:

1) Two horizontally or vertically neighboring key pools share exactly $a|S_c|$ keys[2], where $0 \leq a \leq 0.25$.
2) Two diagonally neighboring key pools share exactly $b|S_c|$ keys, where $0 \leq b \leq 0.25$ and $4a + 4b = 1$.
3) Two non-neighboring key pools share no keys.

We call a and b the overlapping factors. To achieve the above properties, we divide the keys in each key pool into eight partitions (see Fig. 2). Keys in each partition are those keys that are shared between the corresponding neighboring key pools. For example, in Fig. 2, the partition in the upper left corner of E consists of $b \cdot |S_c|$ keys shared between A and E; the partition in the left part of E consists of $a \cdot |S_c|$ keys shared between D and E.

Given the global key pool S and the overlapping factor a and b, we now describe how we can select keys for each key pool $S_{i,j}$ for $i = 1, \ldots, t$ and $j = 1, \ldots, n$. The procedure is also depicted in Fig. 3 for a 4×4 case. First, keys for the first group $S_{1,1}$ are selected from S; then keys for the groups in the first row are selected from S and their left neighbors. Then keys for the groups in the second row to the last row are selected from S and their left, upper-left, upper, and upper-right neighbors. For each row, we conduct the process from left to right. The following procedure describes how we choose keys for each key pool:

1) For group $S_{1,1}$, select $|S_c|$ keys from the global key pool S; then remove these $|S_c|$ keys from S.
2) For group $S_{1,j}$, for $j = 2, \ldots, n$, select $a \cdot |S_c|$ keys from key pool $S_{1,j-1}$; then select $w = (1-a) \cdot |S_c|$ keys from the global key pool S, and remove the selected w keys from S.
3) For group $S_{i,j}$, for $i = 2, \ldots, t$ and $j = 1, \ldots, n$, select $a \cdot |S_c|$ keys from each of the key pools $S_{i-1,j}$ and $S_{i,j-1}$ if they exist; select $b \cdot |S_c|$ keys from each of the key pools $S_{i-1,j-1}$ and $S_{i-1,j+1}$ if they exist; then select w (defined below) keys from the global key pool S, and remove these w keys from S. It is easy to see from the selection procedure that keys selected from the other groups are all distinct.

$$w = \begin{cases} (1-(a+b)) \cdot |S_c|, & \text{for } j = 1 \\ (1-2(a+b)) \cdot |S_c|, & \text{for } 2 \leq j \leq n-1 \\ (1-(2a+b)) \cdot |S_c|, & \text{for } j = n \end{cases}$$

Note that after group G_1 selects s keys ($s = a \cdot |S_c|$ or $s = b \cdot |S_c|$) from its neighbor G_2, no other neighboring groups of G_1 or G_2 can select any one of these s keys, i.e., these s keys are only shared by G_1 and G_2, and no key is shared by more than two neighboring groups in our scheme.

C. Determining $|S_c|$

We calculate the size of the key pool $|S_c|$ for each group, given the size of the global key pool $|S|$.

According to our key pool setup procedure, each group first selects $a|S_c|$ or $b|S_c|$ keys from each of its left, upper, upper-left, and upper-right neighboring groups (if they exist), then selects the rest of the keys from the global key pool S. Fig. 3 depicts the number of keys each group selects from S (each number in the figure should be multiplied by $|S_c|$).

Since keys selected from the other groups are all distinct, the sum of all the numbers in the figure should be equal to $|S|$; therefore we have the following equation:

$$|S_c| = \frac{|S|}{tn - (2tn - t - n)a - 2(tn - t - n + 1)b}$$

For instance, when $|S| = 100,000$, $t = n = 10$, $a = 0.167$, and $b = 0.083$, we have $|S_c| = 1770$. Therefore, the size of the key pool for each group is just 1770.

D. Determining the overlapping factors

The values of the overlapping factors are important to the performance of our scheme. Because we have not introduced the performance metrics yet, we will leave the detailed discussion of the overlapping factors to the next section.

[2]If $a|S_c|$ is not an integer, $\lfloor a|S_c| \rfloor$ should be used instead.

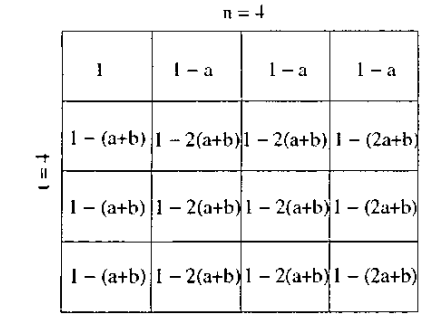


Fig. 3. Key assignment for all the key pools

V. Performance Evaluation

An important goal of this study is to analyze the performance of our proposed scheme. We present both our analytical and simulation results in this section.

A. Evaluation Metrics

We present several criteria that represent desirable characteristics in a key-setup scheme for sensor networks.

- *Connectivity.* We use *global connectivity* to refer to the ratio of the number of nodes in the largest isolated component in the final key-sharing graph to the size of the whole network. If the ratio equals 99%, it means that 99% of the sensor nodes are connected, and the rest 1% are unreachable from the largest isolated component. So, the global connectivity metric indicates the percentage of nodes that are wasted because of their unreachability. We use *local connectivity* to refer to the probability of any two neighboring nodes sharing at least one key. We use p_{local} and p interchangeably to refer to the local connectivity. Both global connectivity and local connectivity are affected by the key pre-distribution scheme.
- *Communication overhead.* Since the probability that two neighboring nodes share a key is less than one, when the two neighboring nodes are not connected directly they need to find a route in the key-sharing graph to connect to each other. We need to determine the number of hops required on this route. Obviously, when the two neighbors are connected directly, the number of hops needed is 1. When more hops are needed to connect two neighboring nodes, the communication overhead of setting up the security association between them is higher. We use $ph(\ell)$ to denote the probability that the smallest number of hops needed to connect two neighboring nodes is ℓ. Obviously, $ph(1)$ equals the local connectivity p_{local}.
- *Resilience against node capture.* We assume that an adversary can mount a physical attack on a sensor node after it is deployed and read secret information from its memory. We need to find how a successful attack on x sensor nodes by an adversary affects the rest of the network. In particular, we want to find the fraction of additional communication (i.e., communications among uncaptured nodes) that an adversary can compromise based on the information retrieved from the x captured nodes.

B. System Configuration

In our analysis and simulations, we use the following setup:

- The size of the key pool, $|S| = 100,000$.
- The number of sensor nodes in the sensor network is $10,000$.
- The deployment area is $1000m \times 1000m$.
- The area is divided into a grid of size $100 = t \times n = 10 \times 10$, with each grid cell of size $100m \times 100m$.
- The center of each grid cell is the deployment point (see Fig. 1(a)).
- The wireless communication range for each node is $R = 40$m.

C. Local Connectivity

We calculate the local connectivity p_{local}, the probability of two neighboring nodes being able to find a common key. Let $B(n_i, n_j)$ be the event that node n_i and node n_j share at least one common key and $A(n_i, n_j)$ be the event that node n_i and node n_j are neighbors. Hence,

$$p_{local} = \Pr(B(n_i, n_j)|A(n_i, n_j)).$$

Note that, since p_{local} is the same for any pair of nodes n_i and n_j, we ignore the node indices n_i and n_j in p_{local}. Let λ be the ratio of the shared key pool between two nodes to $|S_c|$. For example, $\lambda = a$ for groups B and E shown in Fig. 1(a). When the size of the key pool is $|S_c|$, the number of keys shared between two key pools is $\lambda|S_c|$,[3] where the possible values of λ are 1, a, b, and 0.

To calculate Pr(two nodes do not share any key), we use the following strategy: the first node selects i keys from the $\lambda|S_c|$ shared keys, it then selects the remaining $m - i$ keys from the non-shared keys. To avoid sharing any key with the first node, the second node cannot select any of the i keys from those $\lambda|S_c|$ shared keys that are already selected by the first node, so it has to select m keys from the remaining $(|S_c| - i)$ keys from its key pool. Therefore, $p(\lambda)$, the probability that two nodes share at least one key when their key pools have $\lambda|S_c|$ keys in common, can be calculated in the following:[4]

$$\begin{aligned} & p(\lambda) \\ = \; & 1 - \Pr(\text{two nodes do not share any key}) \\ = \; & 1 - \frac{\sum_{i=0}^{min(m,\lambda|S_c|)} \binom{\lambda|S_c|}{i} \binom{(1-\lambda)|S_c|}{m-i} \binom{|S_c|-i}{m}}{\binom{|S_c|}{m}^2}. \end{aligned}$$

We define Ψ as the set of all deployment groups in our scheme. We now consider an infinitesimal rectangular area

[3]For the sake of simplicity, we assume that $\lambda|S_c|$ is an integer; otherwise we could use $\lfloor \lambda|S_c| \rfloor$.

[4]When $\lambda = 1$, $p(\lambda)$ can be simplified to $1 - \frac{\binom{|S_c|-m}{m}}{\binom{|S_c|}{m}}$; when $\lambda = 0$, $p(\lambda) = 0$.

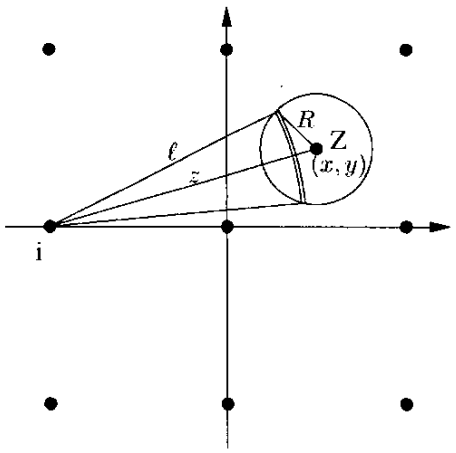


Fig. 4. Probability of nodes residing within a circle ($z > R$).

$dx\,dy$ around position $Z = (x, y)$, as shown in Fig. 4. Based on the two-dimensional Gaussian distribution, the probability that a node n_j from group $j \in \Psi$ with deployment point (x_j, y_j) resides within this small rectangle area is

$$\begin{aligned} & \frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_j)^2+(y-y_j)^2}{2\sigma^2}} \cdot dx\,dy \\ = \; & f_R(d_{jZ}|n_j \in \text{ group } j) \cdot dx\,dy, \end{aligned}$$

where d_{jZ} is the distance between Z and the deployment point of group j, and $f_R(d_{jZ}|n_j \in \text{ group } j)$ is defined as

$$f_R(d_{jZ}|n_j \in \text{ group } j) = \frac{1}{2\pi\sigma^2} e^{-\frac{(d_{jZ})^2}{2\sigma^2}}. \tag{2}$$

When a sensor node n_i resides at position $Z = (x, y)$ as shown in Fig. 4, the probability that the sensor node n_i from group i resides within the circle centered at location Z with radius R is defined as $g(z|n_i \in \text{ group } i)$, where $z = d_{iZ}$, for $i \in \Psi$, is the distance between Z and the deployment point of group i. An example is shown in Fig. 4.

When $z > R$ as shown in Fig. 4,

$$\begin{aligned} & g(z|n_i \in \text{ group } i) \\ = & \int_{z-R}^{z+R} 2\ell \cos^{-1}\left(\frac{\ell^2 + z^2 - R^2}{2\ell z}\right) f_R(\ell|n_i \in \text{ group } i)\, d\ell, \end{aligned}$$

where we have calculated the length of the arc of the ring centered at i and have integrated over all possible values of ℓ.

When $z < R$ as shown in Fig. 5,

$$\begin{aligned} & g(z|n_i \in \text{ group } i) \\ = & \int_0^{R-z} \ell \cdot 2\pi f_R(\ell)\, d\ell \\ + & \int_{R-z}^{z+R} 2\ell \cos^{-1}\left(\frac{\ell^2 + z^2 - R^2}{2\ell z}\right) f_R(\ell|n_i \in \text{ group } i)\, d\ell. \end{aligned}$$

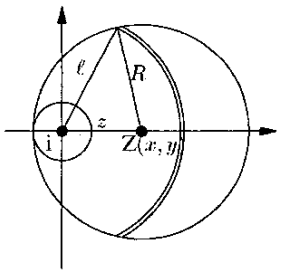


Fig. 5. Probability of nodes residing within a circle ($z < R$).

Thus,

$$\begin{aligned} & g(z|n_i \in \text{ group } i) \\ = \; & 1\{z < R\}\left[1 - e^{-\frac{(R-z)^2}{2\sigma^2}}\right] \\ + & \int_{|z-R|}^{z+R} 2\ell \cos^{-1}\left(\frac{\ell^2 + z^2 - R^2}{2\ell z}\right) f_R(\ell|n_i \in \text{ group } i)\, d\ell, \end{aligned}$$

where $\mathbf{1}\{\cdot\}$ is the set indicator function[5] and $f_R(\ell|n_i \in \text{ group } i)$ is given by Eq. (2).

Assume n_i is a node from group i and n_j is a node from group j, the probability that n_j resides within the rectangle area $dx\,dy$ around point Z and n_i is a neighbor of n_j is the following:

$$f_R(d_{jZ}|n_j \in \text{ group } j) \cdot g(d_{iZ}|n_i \in \text{ group } i) \cdot dx \cdot dy$$

Since the event that node n_i and node n_j share at least one common key is independent of the event that node n_i and node n_j are neighbors, we can calculate the probability that n_j resides within the rectangle area $dx\,dy$ around point Z, and n_i is a neighbor of n_j, and n_i and n_j share at least one common key as:

$$\begin{aligned} & p(\lambda(i,j)) \cdot f_R(d_{jZ}|n_j \in \text{ group } j) \\ & \cdot g(d_{iZ}|n_i \in \text{ group } i) \cdot dx \cdot dy, \end{aligned} \tag{3}$$

where $\lambda(i, j)$ is the ratio of keys shared by the key pool of group i and the key pool of group j:

$$\lambda(i,j) = \begin{cases} 1, & \text{when } i = j; \\ a, & \text{when } i \text{ and } j \text{ are horizontal or} \\ & \text{vertical neighbors;} \\ b, & \text{when } i \text{ and } j \text{ are diagonal neighbors;} \\ 0, & \text{otherwise.} \end{cases}$$

The local connectivity p_{local} is the average of the value in Eq. (3) throughout the entire deployment region, from $(0, 0)$ to (X, Y), and for all the combinations of i and j:

$$\begin{aligned} p_{local} &= \Pr(B(n_i, n_j)|A(n_i, n_j)) \\ &= \frac{\Pr(B(n_i, n_j) \text{ and } A(n_i, n_j))}{\Pr(A(n_i, n_j))}, \end{aligned} \tag{4}$$

[5]The value of $\mathbf{1}\{\cdot\}$ is 1 when the evaluated condition is true, 0 otherwise.

where

$$\Pr(B(n_i, n_j) \text{ and } A(n_i, n_j))$$
$$= \int_{x=0}^{X} \int_{y=0}^{Y} \sum_{j \in \Psi} \sum_{i \in \Psi} \Pr(n_j \in \text{ group } j) \Pr(n_i \in \text{ group } i)$$
$$\cdot f_R(d_{jZ} | n_j \in \text{ group } j)\, g(d_{iZ} | n_i \in \text{ group } i)\, p(\lambda(i,j))\, dx\, dy$$

and

$$\Pr(A(n_i, n_j))$$
$$= \int_{x=0}^{X} \int_{y=0}^{Y} \sum_{j \in \Psi} \sum_{i \in \Psi} \Pr(n_j \in \text{ group } j) \Pr(n_i \in \text{ group } i)$$
$$\cdot f_R(d_{jZ} | n_j \in \text{ group } j)\, g(d_{iZ} | n_i \in \text{ group } i)\, dx\, dy.$$

Since we assume that a sensor node is selected to be in each given group with an equal probability, we have

$$p_{local} = \frac{P_1}{P_2}, \tag{5}$$

where

$$P_1 = \int_{x=0}^{X} \int_{y=0}^{Y} \sum_{j \in \Psi} \sum_{i \in \Psi} f_R(d_{jZ} | n_j \in \text{ group } j)$$
$$\cdot g(d_{iZ} | n_i \in \text{ group } i)\, p(\lambda(i,j))\, dx\, dy,$$

and

$$P_2 = \int_{x=0}^{X} \int_{y=0}^{Y} \sum_{j \in \Psi} \sum_{i \in \Psi} f_R(d_{jZ} | n_j \in \text{ group } j)$$
$$\cdot g(d_{iZ} | n_i \in \text{ group } i)\, dx\, dy.$$

Fig. 6 depicts the local connectivity versus the number of keys (memory usage m) each node carries. We plot both the simulation results and the analytical results calculated from Eq. (5). They match each other very well. We also compare our results with the basic scheme [9]. The figure indicates that our scheme substantially improves local connectivity. For example, with the same setup, when each sensor can carry 100 keys, the local connectivity of the basic scheme is only 0.095; it is improved to 0.687 with deployment knowledge.

D. Global Connectivity

It is possible that the key-sharing graph G in our scheme has a high local connectivity, but G can still have isolated components. Since those components are disconnected, no secure links can be established among them. Therefore, it is important to understand whether G will have too many isolated components. To this end, we measure the global connectivity of the graph G, namely, we measure the ratio of the size of the largest isolated component in G and the size of the whole network. We consider that all the nodes that are not connected to the largest isolated component are useless nodes because they are "unreachable" via secure links.[6]

[6]some of the "unreachable" might be reachable physically because they are within the communication range, but they cannot find a common key with any of the nodes in the largest isolated component.

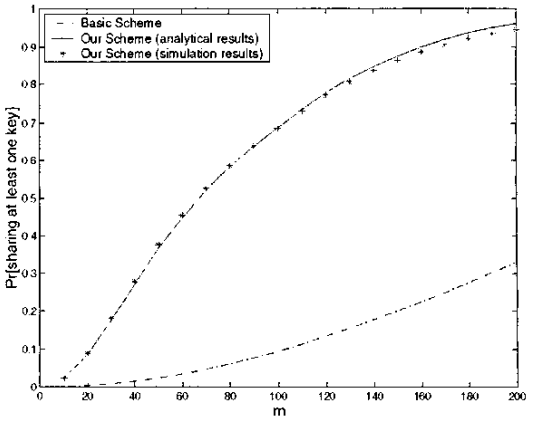


Fig. 6. Local Connectivity: Probability of sharing at least one key between two neighboring nodes.

TABLE I
LOCAL CONNECTIVITY VS. GLOBAL CONNECTIVITY

Memory Usage (m)	Local	Global
10	0.024	0.0132
50	0.383	0.9963
100	0.697	0.9988
120	0.871	0.9997
140	0.871	0.9999
160	0.892	0.9999
180	0.929	0.9999
200	0.956	1.0000

When node distribution and key sharing are uniform, global connectivity can be estimated using the local connectivity and other network parameters using Erdős random graph theorem [23], just like what has been done in [9], [10]. However, since neither our distribution nor our key sharing is uniform, Erdős random graph theorem will not be a good estimation method. Recently, Shakkottai and et. al. have determined the connectivity of a wireless sensor grid network with unreliable nodes [24]. In the future work, we will estimate the global connectivity by using the results given in [24]. In this work, we only use simulation to estimate global connectivity. We use the configuration described in Section V-B to conduct the simulation. The relationships between the memory usage m, the local connectivity, and the global connectivity are shown in Table I. Note that m indicates how many keys each sensor node can store in its memory.

The simulation results indicate that when $m = 100$, only 0.12% of the sensor nodes will be wasted due to the lack of secure links; when $m = 200$, no nodes are wasted. These results have excluded those nodes that are not within the communication ranges of the largest isolated components because they are caused by the deployment, not by our key pre-distribution scheme.

E. Effects of the Overlapping Factors

The values of the overlapping factors are important to the performance of our scheme. For example, when $a = 0.25$

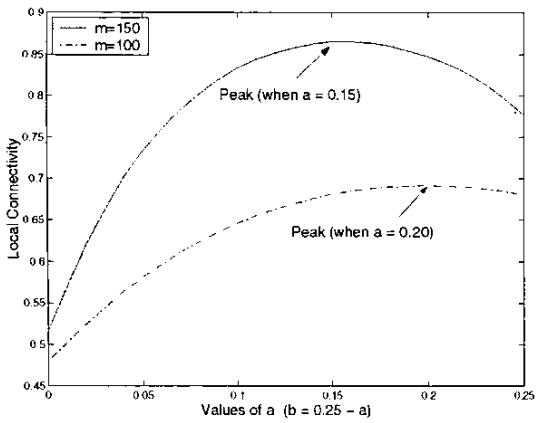


Fig. 7. Local Connectivity vs. Overlapping Factors

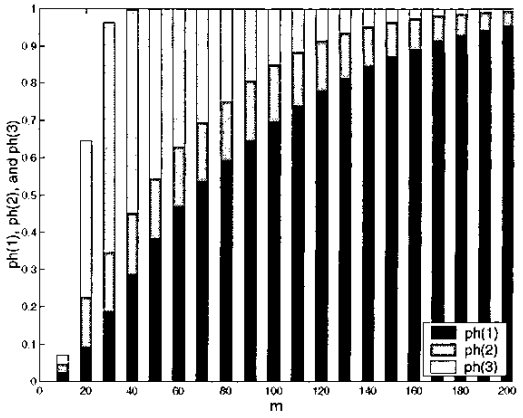


Fig. 8. Communication Overhead

TABLE II
THE BEST OVERLAPPING FACTORS

Memory Usage (m)	a	b
10 - 70	0.25	0
80	0.24	0.01
90	0.22	0.03
100	0.20	0.05
150	0.15	0.10
200	0.14	0.11

and $b = 0$, each group shares keys with its horizontal/vertical neighbors only; when $a = 0$ and $b = 0.25$, each group shares keys with its diagonal neighbors only. The local connectivities for the above two cases are different: when $m = 100$, the local connectivity for the first case is 0.68, whereas for the second case it is 0.48. Therefore, choosing the appropriate combination of a and b is important.

We define the *best overlapping factors* as the combination of a and b that maximizes the local connectivity. They depend on the proportions of different types of neighbors, as well as on the number of keys each sensor node can carry. Fig. 7 depicts the relationship of the local connectivity with the overlapping factors when $m = 100$ and $m = 150$.

Using Eq. (5), we found the best overlapping factors for different values of m. The results are shown in Table II.

F. Communication Overhead

In this subsection, we study the communication overhead of our key pre-distribution scheme when two neighboring nodes cannot find a common key. As we have discussed before, when this situation occurs, these two nodes have to find a path between them in the key-sharing graph G. The shorter the length of the path the better. We use $ph(\ell)$ to denote the probability that the smallest number of hops needed to connect two neighboring nodes is ℓ (note $ph(1) = p_{local}$).

We use simulations to estimate how many of the key setups have to go through ℓ hops, for $\ell = 1, 2, ...$ An analytical approach for estimation similar to that proposed in [11] will be included in our future work. Our results are depicted in Fig. 8. As we can observe from the figure, when each node carries more than 50 keys (i.e. $m > 50$), the sum of $ph(1)$, $ph(2)$, and $ph(3)$ is almost 1, which means that most of the key setups can be conducted within 3 hops.

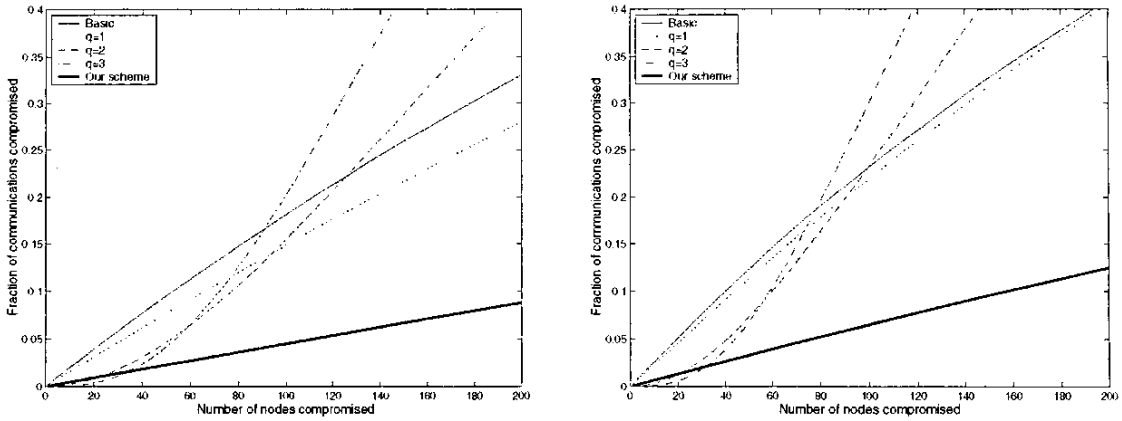
G. Resilience Against Node Capture

To evaluate the resilience of our scheme against node capture, we need to find the fraction of additional communications (i.e., communications among uncaptured nodes) that an adversary can compromise based on the information retrieved from the x captured nodes. Because keys are not uniformly distributed among sensor nodes throughout the entire area, the locations of these x compromised nodes affect the results of our analysis. In this paper, we assume that these x nodes are randomly distributed within the deployment region. We also realize that in reality, these x nodes might not be randomly distributed in the entire region; instead they may have a concentration in a local area because adversaries have a higher probability of compromising nodes around their locations. In that case, the resilience of the network in that local area is lower than that of the entire network. Due to page limits, we leave the local resilience analysis to the extended version of this paper.

Let K be the communication key used for the link between two nodes that are not compromised. When any node other than these two nodes is compromised, the probability that K will not be compromised (i.e. K is not among those keys carried by this compromised node) is $1 - \frac{m}{|S|}$, where m is the number of keys carried by each sensor node. When x nodes are compromised, the probability that K will not be compromised is $(1 - \frac{m}{|S|})^x$. Therefore, the expected fraction of total keys being compromised can be estimated as:

$$1 - (1 - \frac{m}{|S|})^x. \quad (6)$$

The results and comparison with existing key pre-distribution schemes are depicted in Fig. 9 ("Basic" refers to the basic Eschenauer-Gligor scheme; "$q = 1, 2, 3$" refers to the Chan-Perrig-Song scheme). The figures show that our scheme substantially lowers the fraction of compromised



(a) $p = 0.33$ (b) $p = 0.50$

Fig. 9. Network Resilience: Comparisons with existing schemes

communication after x nodes are compromised. The most important reason for such an improvement is that, to achieve the same local connectivity while using the same key pool size $|S|$, our scheme requires a much smaller m. For example, when $|S| = 100,000$, to achieve $p = 0.33$, the basic scheme needs $m = 200$, but our scheme only needs $m = 46$; to achieve $p = 0.50$ the basic scheme needs $m = 263$, but our scheme only needs $m = 67$. It is easy to see from Eq. (6), the smaller the value of m is, the better the resilience. Such an improvement is attributed to the deployment knowledge, which enables us to reduce the number of unnecessary keys carried by each sensor node.

VI. Conclusions and Future Work

We have described a random key pre-distribution scheme that uses deployment knowledge. With such knowledge, each node only needs to carry a fraction of the keys required by the other key pre-distribution schemes [9], [10] while achieving the same level of connectivity. The reduction in memory usage not only relieves the memory requirement on the memory-constrained sensor node, but more importantly, it substantially improves network's resilience against node capture. We have shown these improvements using our analytical and simulation results.

Having demonstrated the dramatic improvement in the performance of the Eschenauer-Gligor scheme, in our future work, we will investigate how much the deployment knowledge can improve the q-composite random key pre-distribution scheme and the pairwise key pre-distribution scheme proposed by Chan, Perrig, and Song [10]. In addition, we will study the global connectivity, communication overhead, and the local resilience as we mentioned in the last section. Other deployment strategies and associated distributions will also be considered.

VII. Acknowledgment

The authors acknowledge supports from the United States National Science Foundation IIS-0219560, IIS-0312366, DUE-0231122, the SUPRIA program of the CASE Center at Syracuse University, and the National Science Council of Taiwan, R.O.C., under grants NSC 90-2213-E-260-007 and NSC 91-2213-E-260-021. The authors would also like to thank all the anonymous reviewers for their valuable comments.

References

[1] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next century challenges: Mobile networking for smart dust," in *Proceedings of the 5th Annual ACM/IEEE Internation Conference on Mobile Computing and Networking (MobiCom)*, 1999, pp. 483–492.

[2] Wireless Integrated Network Sensors, University of California, Available: http://www.janet.ucla.edu/WINS.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, August 2002.

[4] B. C. Neuman and T. Tso, "Kerberos: An authentication service for computer networks," *IEEE Communications*, vol. 32, no. 9, pp. 33–38, September 1994.

[5] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, pp. 644–654, November 1976.

[6] 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, 1978.

[7] A. Perrig, R. Szewczyk, V. Wen, D. Cullar, and J. D. Tygar, "Spins: Security protocols for sensor networks," in *Proceedings of the 7th Annual ACM/IEEE Internation Conference on Mobile Computing and Networking (MobiCom)*, Rome, Italy, July 2001, pp. 189–199.

[8] R. Anderson and M. Kuhn, "Tamper resistance - a cautionary note," in *Proceedings of the Second Usenix Workshop on Electronic Commerce*, November 1996, pp. 1–11.

[9] 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*, Washington, DC, USA, November 18-22 2002, pp. 41–47.

[10] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *IEEE Symposium on Security and Privacy*, Berkeley, California, May 11-14 2003, pp. 197–213.

[11] 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)*, Washington, DC, USA, October 27-31 2003, pp. 42–51.

[12] 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)*, Washington, DC, USA, October 27-31 2003, pp. 52–61.

[13] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences," *Lecture Notes in Computer Science*, vol. 740, pp. 471–486, 1993.

[14] M. Tatebayashi, N. Matsuzaki, and D. B. Newman, "Key distribution protocol for digital mobile communication systems," *Advances in Cryptology - CRYPTO'89*, pp. 324–334, 1989, INCS Volume 435, Springer-verlag.

[15] C. Park, K. Kurosawa, T. Okamoto, and S. Tsujii, "On key distribution and authentication in mobile radio networks," *Advances in Cryptology - EuroCrypt'93*, pp. 461–465, 1993, INCS Volume 765, Springer-verlag.

[16] M. Beller and Y. Yacobi, "Fully-fledged two-way public key authentication and key agreement for low-cost terminals," *Electronics Letters*, vol. 29, no. 11, pp. 999–1001, 1993.

[17] C. Boyd and A. Mathuria, "Key establishment protocols for secure mobile communications: A selective survey," *Lecture Notes in Computer Science*, vol. 1438, pp. 344–355, 1998.

[18] L. Zhou and Z. J. Haas, "Securing ad hoc networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, 1999.

[19] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing robust and ubiquitous security support for mobile ad-hoc networks," in *International Conference on Network Protocols (ICNP)*, 2001, pp. 251–260.

[20] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *7th International Workshop on Security Protocols*, vol. 1796, 1999, pp. 172–194, INCS Volume 1796, Springer-verlag.

[21] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, 2nd ed. Reading, MA: Addison-Wesley Publishing Company, Inc., 1994.

[22] C. E. Perkins, Ed., *Ad Hoc Networking*. Addison-Wesley, 2001.

[23] Erdős and Rényi, "On random graphs I," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.

[24] S. Shakkottai, R. Srikant, and N. Shroff, "Unreliable sensor grids: coverage, connectivity and diameter," in *Proceedings of the IEEE INFOCOM*, 2003, pp. 1073–1083.

0-7803-8355-9/04/$20.00 ©2004 IEEE.