1-1-2006

A key predistribution scheme for sensor networks using deployment knowledge

Wenliang Du
Syracuse University, wedu@ecs.syr.edu

Jing Deng
University of New Orleans

Yunghsiang S. Han
National Taipei University

Pramod K. Varshney
Syracuse University, varshney@ecs.syr.edu

A Key Pre-distribution Scheme for Sensor Networks Using Deployment Knowledge

Wenliang Du*, Jing Deng†, Yunghsiang S. Han‡, and Pramod K. Varshney*
*Department of Electrical Engineering and Computer Science
Syracuse University, Syracuse, NY 13244-1240, USA
Email: {wedu, varshney}@ecs.syr.edu
†Department of Computer Science
University of New Orleans, New Orleans, LA 70148, USA
Email:jing@cs.uno.edu
‡Department of Computer Science and Information Engineering
National Taipei University, Taiwan, R.O.C.
Email: yshan@mail.ntpu.edu.tw

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, and short-range radio communication capabilities. 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,

This paper is currently under submission to the IEEE Transactions on Dependable and Secure Computing. This paper is an extended version of the conference paper, *A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge*, published in Proceedings of the IEEE INFOCOM, March 7-11, 2004, Hong Kong. Pages 586-597.

impersonate one of the network nodes (in this paper, we use the terms sensors, sensor nodes, and nodes interchangeably), 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 be in the same neighborhood before deployment, keys can be decided *a priori*. However, because of the randomness of deployment, it might be infeasible to learn the set of neighbors *a priori*.

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 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 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 this key as their shared secret key [9]. An overview of this scheme is given in Section III. The Eschenauer-Gligor scheme has further been 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 [9]–[12] provided viable solutions to the key pre-distribution problem, they have not exploited an important 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 to improve the performance of key pre-distribution.

Knowing which sensors are close to each other is important for 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*. If we know perfectly the neighbors of each node in the network, key pre-distribution becomes trivial: for each node n_i, we just need to generate a pairwise key between n_i and each of its neighboring nodes, and save these keys in n_i'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. Still, the number of possible neighbors can be very large and it may not be feasible for a sensor to store one secret key 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 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 increased 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. In this paper, we look at non-uniform pdfs, which imply that we know that a sensor is more likely to be deployed in certain areas. We will show how this knowledge can help to improve the random key pre-distribution scheme proposed by Eschenauer and Gligor in [9] and the scheme proposed by Du, Deng, Han, and Varshney in [11]. To demonstrate the effectiveness of our method, we have studied a specific distribution, the Normal (Gaussian) distribution, in great depth. Our results show substantial improvement over existing schemes that do not exploit deployment knowledge.

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. We are the first to attempt the use of deployment knowledge in key pre-distribution.
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.

II. Related Work

The Eschenauer-Gligor scheme [9] has been briefly described earlier in Section I. We will give a more detailed description of this scheme in Section III. Based on the Eschenauer-Gligor scheme, Chan, Perrig, and Song proposed a q-composite random key pre-distribution scheme [10]. The major 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 improved 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

nodes other than the compromised ones are affected is close to zero. This desirable property lowers the initial payoff of small-scale network breaches to an adversary, and makes it necessary for the adversary to attack a significant portion of the network. A similar method was also developed by Liu and Ning [12].

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. Chan and Perrig proposed PIKE, a class of key-establishment protocols that involves using one or more sensor nodes as a trusted intermediary to facilitate key establishment [13]. Anderson, Chan, and Perrig also studied how the key distribution problem can be dealt with in environments with a partially present, passive adversary [14].

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 [15]. These schemes focus on saving communication costs while memory constraints are not placed on group members.

Several other key distribution schemes have been proposed for mobile computing, although they are not specifically targeted at sensor networks. Tatebayashi, Matsuzaki, and Newman considered key distribution for resource-starved devices in a mobile environment [16]. This work is further improved by Park et al. [17]. Other key agreement and authentication protocols include the one by Beller and Yacobi [18]. A survey on key distribution and authentication for resource-starved devices in mobile environments is given in [19]. 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 to secure ad hoc network using secret sharing and threshold cryptography [20]. Kong et al. also propose localized public-key infrastructure mechanisms, based on secret sharing schemes [21].

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 [22]. Key pre-distribution is similar to the "imprinting" process, but their objectives are different.

III. Background

A. The Eschenauer-Gligor (EG) Scheme

The *Eschenauer-Gligor scheme* (referred to as the basic scheme or the EG scheme hereafter) 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 τ distinct cryptographic keys from a key pool S, and stores them in its memory. This set of τ 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 τ 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 graph (called *key 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 key 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 $\tau = 200$ 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 use a large key pool while still maintaining high connectivity and the same memory usage? In this paper, we use deployment knowledge to solve this problem.

B. The Du-Deng-Han-Varshney (DDHV) Scheme

Blom proposed a key pre-distribution method that allows any pair of nodes in a network to be able to derive a pairwise secret key [23]. It has the property that as long as no more than λ nodes are compromised, all communication links of non-compromised nodes remain secure (we refer to this as being "λ-secure"). We now briefly describe Blom's scheme (we have made some slight modifications to the scheme in order to make it more suitable for sensor networks, but the essential features remain unchanged).

We assume some agreed-upon $(\lambda + 1) \times N$ matrix G over a finite field $GF(q)$, where N is the size of the network and $q > N$. This matrix G is public information and may be shared by different systems; even adversaries are allowed to know G. During the key generation phase the base station creates a random $(\lambda + 1) \times (\lambda + 1)$ symmetric matrix D over $GF(q)$, and computes an $N \times (\lambda + 1)$ matrix $A = (D \cdot G)^T$, where $(D \cdot G)^T$ is the transpose of $D \cdot G$. Matrix D must be kept secret, and should not be disclosed to adversaries or to any sensor nodes (although, as will be discussed, one row of $(D \cdot G)^T$ will be disclosed to each sensor node). Because D is symmetric, it is easy to see that

$$\begin{aligned} A \cdot G &= (D \cdot G)^T \cdot G = G^T \cdot D^T \cdot G = G^T \cdot D \cdot G \\ &= (A \cdot G)^T; \end{aligned}$$

i.e., $A \cdot G$ is a symmetric matrix. If we let $K = A \cdot G$, we know that $K_{ij} = K_{ji}$, where K_{ij} is the element in the ith row and jth column of K. The idea is to use K_{ij} (or K_{ji}) as the pairwise key between node i and node j. Figure 1 illustrates how the pairwise key $K_{ij} = K_{ji}$ is generated. To carry out the above computation, nodes i and j should be able to compute K_{ij} and K_{ji}, respectively. This can be easily achieved using the following key pre-distribution scheme, for $k = 1, \ldots, N$:

1) store the kth row of matrix A at node k, and
2) store the kth column of matrix G at node k.[1]

Then, when nodes i and j need to establish their pairwise key, they first exchange their columns of G and then compute K_{ij} and K_{ji}, respectively, using their private rows of A. Because G is public information, its columns can be transmitted in plaintext. It has been shown [23] that the above scheme is λ-secure if any $\lambda + 1$ columns of G are linearly independent. This λ-secure property guarantees that no coalition of up to λ nodes (not including i and j) have any information about K_{ij} or K_{ji}.

We define the set of keys generated from A and G as a *key space*. According to the Blom scheme, if any two nodes carry their corresponding information from the same key space, they can find a common key between themselves. Roughly speaking, Blom's scheme uses a *single* key space. By changing the values of matrices D and G, we can create different key spaces.

Motivated by the random key pre-distribution schemes [9], [10], Du et al. developed an improved key pre-distribution scheme using *multiple* key spaces (we call it the DDHV scheme) [11]. The DDHV scheme

[1] In practice, sensors need not store the whole column, because each column can be generated from a single field element [11].

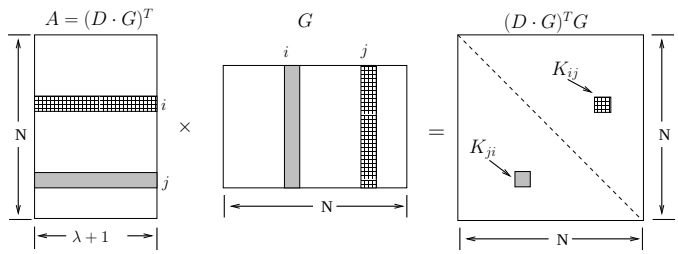


Fig. 1. Generating keys in Blom's scheme.

first constructs ω spaces using Blom's scheme, and then have each sensor node carry key information from τ (with $2 \leq \tau < \omega$) randomly selected key spaces. Now (from the properties of the underlying Blom scheme), if two nodes carry key information from a common space they can compute a shared key. Of course, unlike Blom's scheme, it is no longer certain that two nodes can generate a pairwise key; instead (as in the Eschenauer-Gligor random key pre-distribution scheme), such a connectivity is probabilistic.

It should be noted that when a key space has $\lambda = 0$, a compromise of one (i.e. $\lambda + 1$) node from this key space will compromise the entire key space. This is equivalent to having one key in this key space. Therefore, by letting $\lambda = 0$, each key space collapses to one key, and thus the DDHV scheme reduces to the EG scheme. From this perspective, the EG scheme is actually a special case of the DDHV scheme. Therefore, in this paper, we focus only on the DDHV scheme.

IV. Modeling of the Deployment Knowledge

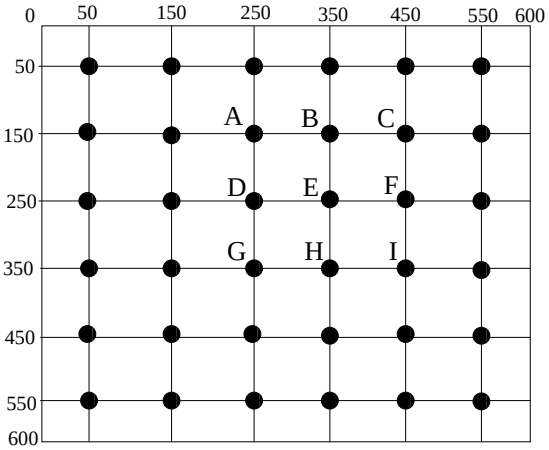
We assume that sensor nodes are static once they are deployed. We define *deployment point* as the desired point where a sensor is to be deployed. This is not likely the location where the sensor resides eventually. The sensor node can reside at points around this desired point according to a certain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

A. Group-based Deployment Model

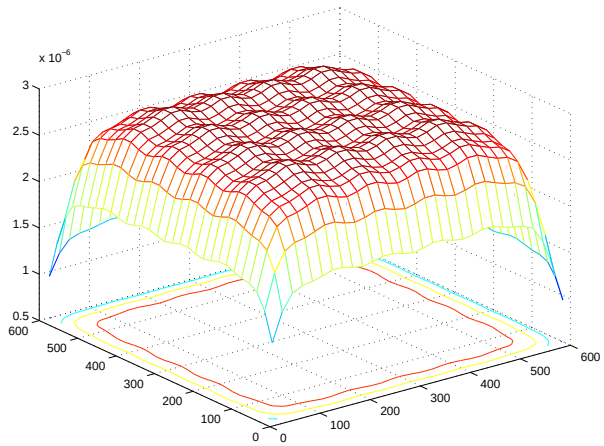
In practice, it is quite common that nodes are deployed in groups, i.e., a group of sensors are deployed at a single deployment point, and the pdfs of the final resident points of all the sensors in each batch (or group) are the same. In this work, we assume such a group-based deployment, and we model the deployment knowledge as follows (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 develop 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 $f(x, y|k \in G_{i,j})$. An example of the pdf is a two-dimensional Gaussian distribution.

When $f(x, y|k \in G_{i,j})$ is a uniform distribution over the deployment region for all $G_{i,j}$'s, we do not know which nodes are more likely to be close to each other *a priori* because the resident point of a



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



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

Fig. 2. Node Deployment

node can be anywhere within the region with the same probability. However, when $f(x, y|k \in G_{i,j})$ 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 Eschenauer-Gligor random key pre-distribution scheme [9] and the DDHV scheme [11], when the size of the key-space pool S becomes smaller, connectivity increases. Since these schemes assume no deployment knowledge (i.e. the distribution $f(x, y|k \in G_{i,j})$ is uniform), every node should choose from the same key-space pool because they are equally likely to be neighbors. However, as we have discussed, when the function $f(x, y|k \in G_{i,j})$ 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-space pools should be different, rather than the same global key-space pool S.

We use $S_{i,j}$ to represent the key-space 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 $S_{i,j}$ (for the sake of simplicity, we let all $S_{i,j}$'s have 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_{i_1,j_1} and G_{i_2,j_2} are farther away from each other, the amount of overlap between S_{i_1,j_1} and S_{i_2,j_2} becomes smaller or zero.

B. 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. Our key pre-distribution scheme is for the most part model independent. We propose our scheme in a manner whereby it can be instantiated to use other deployment models. To keep the presentation concrete,

we use an example model; namely, we model the sensor deployment distribution as a two-dimensional Gaussian distribution (also called Normal distribution), Our methodology should be easily adaptable to other deployment models.

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 [24]:

$$f(x, y|k \in G_{i,j}) \quad = \quad \frac{1}{2\pi\sigma^2} e^{-[(x-x_i)^2+(y-y_j)^2]/2\sigma^2}. \tag{1}$$

Although the distribution function for each single group is non-uniform, we still prefer 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, $\frac{1}{t \cdot n}$, the average deployment distribution (pdf) of any sensor node over the entire region is:

$$f_{overall}(x, y) = \frac{1}{t \cdot n} \cdot \sum_{i=1}^{t} \sum_{j=1}^{n} f(x, y|k \in G_{i,j}). \tag{2}$$

To see the overall distribution of sensor nodes over the entire deployment region, we have plotted $f_{overall}$ in Eq. (2) for $6 \times 6 = 36$ groups over a $600m \times 600m$ square region with the deployment points $2\sigma = 100m$ apart (assuming $\sigma = 50$). Figure 2(a) shows all the deployment points, and Figure 2(b) shows the overall pdf. From Figure 2(b), we can see that the pdf is almost flat (i.e. nodes are fairly evenly distributed) in the whole region except near the boundaries.

V. 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. This new scheme is based on the original DDHV scheme, so we call it the DDHV-D scheme.[2] In this scheme, we assume that the sensor nodes are evenly divided into $t \times n$ groups $G_{i,j}$, for $i = 1, \ldots, t$, and $j = 1, \ldots, n$. We assume that the global key-space pool is S with size $|S|$, and also assume that the deployment points are arranged in a grid depicted in Figure 2(a). Each node carries τ key spaces.

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 DDHV scheme [11], but because of deployment knowledge, the first phase is considerably different.

Step 1: Key Pre-distribution phase. This phase is conducted offline and before the sensors are deployed. First we need to divide the key-space pool S into $t \times n$ key-space 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-space 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-space pools $S_{i,j}$ is to allow the nearby key-space pools to share more key spaces, while those far away from each other share fewer key spaces or no key space at all. Steps for setting up key-space pools will be discussed in details later.

[2] "D" after the hyphen indicates the use of deployment knowledge.

After the key-space pools are set up, for each sensor node in the deployment group $G_{i,j}$, we randomly select τ key spaces from its corresponding key-space pool $S_{i,j}$; then for each selected key space, we load the corresponding row of its matrix (i.e. matrix A) into the memory of the node.

Step 2: Shared-key discovery phase. After deployment, each node needs to discover whether it shares any key space with its neighbors. To do this, each node broadcasts a message containing the indices of the key spaces it carries. Each neighboring node can use these broadcast messages to find out if there exists a common key space it shares with the broadcasting node. If such a key space exists, using the Blom scheme, the two neighboring nodes can derive a pair-wise key from the common key space, and use the key to secure the communication channel between themselves.

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

> *Definition 1:* (Key-Space Sharing Graph) Let V represent all the nodes in the sensor network. A Key-Space Sharing Graph $GKS(V, E)$ 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 space, 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 key space 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 space could still come up with a secret key between them. The idea is to use the secure channels that have already been established in the key-space sharing graph GKS: as long as the graph is connected, two neighboring nodes i and j can always find a path in GKS from i to j. Assume that the path is i, v_1, ..., v_h, 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_h. 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 [25], a common technique used in multi-hop 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.

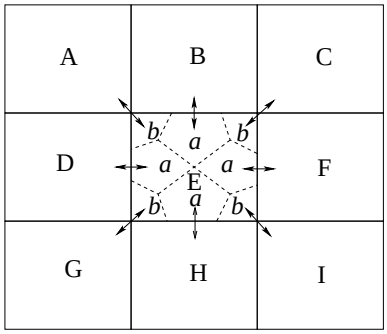
B. Setting Up Key-Space Pools

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

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

We call a and b the overlapping factors. To achieve the above properties, we divide the key spaces in each key-space pool into eight partitions (see Figure 3(a)). Key spaces in each partition are those

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



(a) Shared key spaces between neighboring key-space pools

n = 4

t = 4

1	1 − a	1 − a	1 − a
1 − (a+b)	1 − 2(a+b)	1 − 2(a+b)	1 − (2a+b)
1 − (a+b)	1 − 2(a+b)	1 − 2(a+b)	1 − (2a+b)
1 − (a+b)	1 − 2(a+b)	1 − 2(a+b)	1 − (2a+b)

(b) Key space assignment for all the key-space pools

Fig. 3. Key-Space Pools

key spaces that are shared between the corresponding neighboring key-space pools. For example, in Figure 3(a), the partition in the upper left corner of E consists of $b \cdot |S_c|$ key spaces shared between A and E; the partition in the left part of E consists of $a \cdot |S_c|$ key spaces shared between D and E.

Given the global key-space pool S and the overlapping factor a and b, we now describe how we can select key spaces for each key-space pool $S_{i,j}$ for $i = 1, \ldots, t$ and $j = 1, \ldots, n$. The procedure is also depicted in Figure 3(b) for a 4×4 case. First, key spaces for the first group $S_{1,1}$ are selected from S; then key spaces for the groups in the first row are selected from S and their left neighbors. Then key spaces 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 key spaces for each key-space pool:

1) For group $S_{1,1}$, select $|S_c|$ key spaces from the global key-space pool S; then remove these $|S_c|$ key spaces from S.
2) For group $S_{1,j}$, for $j = 2, \ldots, n$, select $a \cdot |S_c|$ key spaces from the key-space pool $S_{1,j-1}$; then select $w = (1 - a) \cdot |S_c|$ key spaces from the global key-space pool S, and remove the selected w key spaces from S.
3) For group $S_{i,j}$, for $i = 2, \ldots, t$ and $j = 1, \ldots, n$, select $a \cdot |S_c|$ key spaces from each of the key-space pools $S_{i-1,j}$ and $S_{i,j-1}$ if they exist; select $b \cdot |S_c|$ key spaces from each of the key-space pools $S_{i-1,j-1}$ and $S_{i-1,j+1}$ if they exist; then select w (defined below) key spaces from the global key-space pool S, and remove these w key spaces from S.

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

Note that after any group (e.g., G_1) selects s key spaces ($s = a \cdot |S_c|$ or $s = b \cdot |S_c|$) from its neighbor (e.g., G_2), no other neighboring groups of G_1 or G_2 can select any one of these s key spaces, i.e., these s key spaces are only shared by G_1 and G_2. In other words, no key space is shared by more than two neighboring groups in our scheme. Although this requirement is not necessary in practice, it significantly simplifies our analysis.

VI. Performance and Security: Analytical Results

In this section we analyze the performance and security of our scheme. We present our analytical results on the following two metrics:

- *Local connectivity*. We use local connectivity to refer to the probability of any two neighboring nodes sharing at least one key space. We use p_{local} and p interchangeably to refer to the local connectivity. The local connectivity directly affects the performance of the scheme.
- *Resilience against node capture*. In a hostile environment, adversary can mount physical attacks 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 communications (i.e., communications among uncaptured nodes) that an adversary can compromise based on the information retrieved from the x captured nodes.

A. Computing Local Connectivity

We randomly pick any two nodes u and v in the network. Let $A(u,v)$ be the event that u and v are neighbors; let $B(u,v)$ be the event that u and v share at least one common key space. Therefore, the local connectivity p_{local} (i.e., the probability of two neighboring nodes being able to find a common key space), is the following conditional probability:

$$p_{local} = \Pr(B(u,v) \mid A(u,v)) = \frac{\Pr(B(u,v) \text{ and } A(u,v))}{\Pr(A(u,v))}. \tag{3}$$

Since u and v are picked randomly, the above probability is the average over all possible pairs of nodes. Defining Ψ as the set of all deployment groups in our scheme, we have

$$\begin{aligned}\Pr(A(u,v)) &= \sum_{j\in\Psi}\sum_{i\in\Psi}\Pr(A(u,v) \mid u\in G_i \text{ and } v\in G_j)\cdot\Pr(u\in G_i \text{ and } v\in G_j)\\ &= \frac{1}{(n\cdot t)^2}\sum_{j\in\Psi}\sum_{i\in\Psi}\Pr(A(u,v) \mid u\in G_i \text{ and } v\in G_j).\end{aligned}$$

Note that in the above equation, because the two nodes u and v are selected independently, and each of them is selected to be in any given deployment group with an equal probability, we have $\Pr(u \in G_i \text{ and } v\in G_j) = \frac{1}{(nt)^2}$, where $n\cdot t$ is the number of deployment groups. Similar to the above equation for $\Pr(A(u,v))$, we have the following equation:

$$\begin{aligned}&\Pr(B(u,v) \text{ and } A(u,v))\\ =\ &\sum_{j\in\Psi}\sum_{i\in\Psi}\Pr(B(u,v) \text{ and } A(u,v) \mid u\in G_i \text{ and } v\in G_j)\cdot\Pr(u\in G_i \text{ and } v\in G_j)\\ =\ &\frac{1}{(nt)^2}\sum_{j\in\Psi}\sum_{i\in\Psi}\Pr(B(u,v) \text{ and } A(u,v) \mid u\in G_i \text{ and } v\in G_j).\end{aligned}$$

Because events $B(u,v \mid u\in G_i \text{ and } v\in G_j)$ and $A(u,v \mid u\in G_i \text{ and } v\in G_j)$ are independent, [4] we

[4]Note that unconditional events $B(u,v)$ and $A(u,v)$ are not independent, because they both depend on the deployment groups that u and v come from.

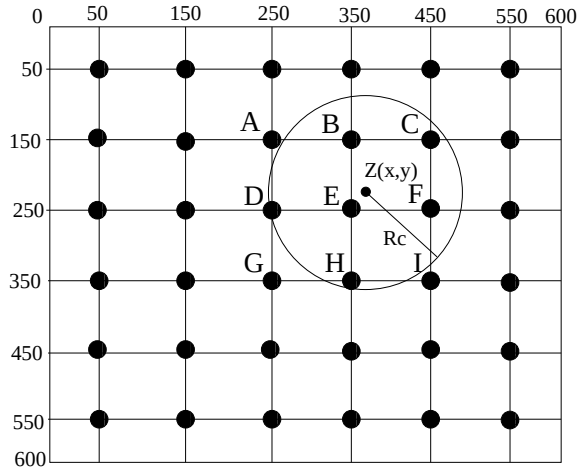


Fig. 4. Resilience against node capture. An attack circle, centered at $Z(x, y)$ with radius R_c, is shown in the network. The adversary randomly picks x_c nodes from this circle.

have the following:

$$\begin{aligned} & \Pr(B(u,v) \text{ and } A(u,v)) \\ = \; & \frac{1}{(nt)^2} \sum_{j \in \Psi} \sum_{i \in \Psi} \Pr(B(u,v) \mid u \in G_i \text{ and } v \in G_j) \cdot \Pr(A(u,v) \mid u \in G_i \text{ and } v \in G_j). \end{aligned}$$

Therefore, to compute the local connectivity, we just need to compute $\Pr(A(u,v) \mid u \in G_i$ and $v \in G_j)$ and $\Pr(B(u,v) \mid u \in G_i$ and $v \in G_j)$. To simplify notations, we use n_i to replace u and n_j to replace v; the subscripts i and j indicate that n_i is from G_i and n_j is from G_j. We can therefore omit the condition $(u \in G_i$ and $v \in G_j)$ in our notation. The probability of local connectivity in Eq. (3) becomes

$$p_{local} = \frac{\sum_{j \in \Psi} \sum_{i \in \Psi} \Pr(B(n_i, n_j)) \cdot \Pr(A(n_i, n_j))}{\sum_{j \in \Psi} \sum_{i \in \Psi} \Pr(A(n_i, n_j))} \; . \tag{4}$$

Therefore, we need to compute $\Pr(A(n_i, n_j))$ and $\Pr(B(n_i, n_j))$ in order to find p_{local}. The detailed derivations of these two are given in Appendices I and II, with $\Pr(A(n_i, n_j))$ given by Equation (8) and $\Pr(B(n_i, n_j))$ given by Equation (10). It should be noted that $\Pr(A(n_i, n_j))$ solely depends on the deployment model, while $\Pr(B(n_i, n_j))$ solely depends on the key pre-distribution.

B. Resilience Analysis

In order to analyze the resilience of the DDHV-D scheme, we need to have a model for the adversary's attacks. While establishing such models, we consider a realistic scenario in which the adversary intrudes a region inside the sensor network and randomly captures and compromises x_c sensors within this region. We explain the attack model in the following:

- We assume that the adversary captures nodes randomly within a region;
- The region is assumed to be a circle[5] centered at point $Z(x, y)$ with radius R_c. We term such circle as the *attack circle* and call R_c the *attack radius*. An example of an attack circle is shown

[5]The analysis of other shapes is similar, albeit with more complicated formulas.

in Figure 4. Note that when the circle is large enough to contain the entire deployment region, the attack model reduces to the uniform-random attack, in which the probability that any node in the entire deployment region is compromised is the same.

Under this attack model, we analyze the resilience of our key pre-distribution scheme. We explore the effect of the capture of x_c sensor nodes by an adversary on the security of the rest of the network. In particular, we calculate the fraction of additional communication (i.e., communication among the uncaptured nodes) that an adversary can compromise based on the information retrieved from the x_c captured nodes. To compute this fraction, we first compute the probability that any one of the additional communication links is compromised after x_c nodes are captured. Note that we only consider the links in the key-space-sharing graph, and each of these links is secured using a key computed from the common key space shared by the two nodes of this link.

Before we present our detailed analysis on resilience, we summarize our approach in the following for the benefit of clarity: based on the above assumptions, we can calculate, among all sensors in the attack circle, the average number of sensors that are deployed from each specific group. Since the adversary compromises x_c sensors randomly inside the circle, the average number of compromised sensors that are deployed from the specific group can be derived. Based on the key pool sharing technique shown in Figure 3(b), we derive the average number of sensors that are compromised and are carrying keys from the same key pool. Then we use the method in [11] to calculate the fraction of additional communication that an adversary can compromise based on the information retrieved from the x_c captured nodes.

Let z_i denote the distance between the deployment point of group G_i and location Z, the center of the attack circle (cf. Figure 4). Let $g_i = g(z_i \mid n_i \in G_i)$ represent the probability that a sensor node n_i from group G_i resides within the attack circle. The details of the derivation for g_i is given in Appendix I, and the results are given in Equations (6) and (7).

With N sensors divided into $t \times n$ groups, each group has $\frac{N}{t \times n}$ sensors. The expected number of sensors that are from group G_i and reside in the attack circle is

$$N_i = \frac{N}{t \cdot n} g_i \ ,$$

with the expected number of total sensors in the attack circle center at $Z(x, y)$ as

$$N(Z(x, y), R_c) = \sum_{i \in \Psi} N_i = \sum_{i \in \Psi} \frac{N}{t \cdot n} g_i \ .$$

Since the adversary randomly chooses x_c sensors among these $N(Z(x, y), R_c)$ sensors, the expected number of captured sensors that are deployed from group G_i is

$$x_i(x, y, R_c) = x_c \cdot \frac{N_i}{N(Z(x, y), R_c)} = x_c \cdot \frac{g_i}{\sum_{j \in \Psi} g_j} \ .$$

Next, we look for the expected number of sensors that draw their keys from the same group of key spaces (from group G_i). Since the sensors that are deployed from the neighboring groups of G_i share some key spaces with this group G_i, we need to count the weighted sum of the numbers of nodes that have been captured from all these groups:

$$X_i(x, y, R_c) = \sum_{j \in \Psi_i} \frac{\xi(i, j)}{|S_c|} \cdot x_j(x, y, R_c) \ ,$$

where Ψ_i represents i and the indices of all neighboring groups of group i, and $\xi(i, j)$, given by Equation (9), is the number of common key spaces shared by the key pools of groups G_i and G_j. For example, $\Psi_i = \{A, \cdots, I\}$ when $i = E$ in Figure 4.

Let A be the deployment area, c the link between u and v, and $C(x, y)$ the event that the attack circle is centered at (x, y). Due to the fact that $C(x, y)$ is independent of $A(u, v)$ and $B(u, v)$, we have

$$\begin{aligned} & \Pr(c \text{ is compromised} \mid A(u,v) \text{ and } B(u,v)) \\ = \; & \frac{1}{A}\int_A \Pr(c \text{ is compromised} \mid C(x,y) \text{ and } A(u,v) \text{ and } B(u,v))\; dxdy. \end{aligned}$$

The derivation of $\Pr(c \text{ is compromised} \mid C(x, y) \text{ and } A(u, v) \text{ and } B(u, v))$ is rather cumbersome. Therefore, we move it to Appendix III (cf. Equation (13)).

VII. Performance and Security Evaluation: Numerical Results

An important goal of this study is to understand the performance of the DDHV-D scheme. However, because of the complexity of the analytical results obtained for local connectivity and resilience, it is difficult to understand the performance from the equations that we have derived. In this section, we present numerical results corresponding to those derived equations. We show the performance of the DDHV-D scheme as well as the comparisons with the existing key pre-distribution schemes. More importantly, we will use the numerical results to understand the relationships among the parameters λ, memory usage m, local connectivity p_{local}, and resilience, as their relationships are difficult to understand from the rather complicated analytical results. Note that m is defined in units of key size; namely, if each key is 64 bits long, then the total amount of memory usage is $64 \cdot m$ bits. The relationship between the memory usage m and the number (τ) of key spaces each sensor can carry is the following [11]:

$$\tau = \lfloor \frac{m}{\lambda + 1} \rfloor.$$

A. System Configuration

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

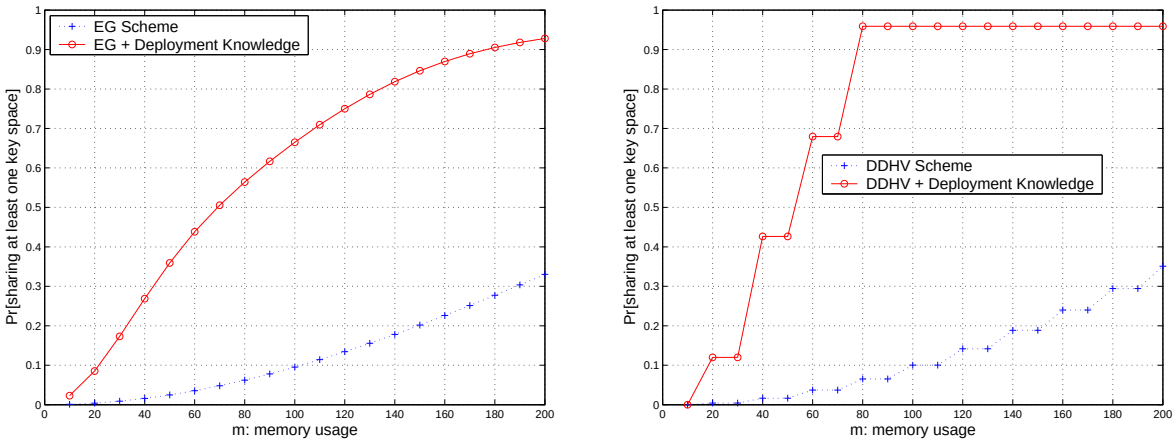
- The number of sensor nodes in the sensor network is $10,000$.
- The deployment area is $1000m \times 1000m$.
- The area is divided into a grid of size $100 = t \times n = 10 \times 10$, with each grid cell of size $100m \times 100m$.
- The center of each grid cell is the deployment point (see Figure 2(a)).
- The wireless communication range for each node is $R = 40$m.
- We assume that the node deployment follows a two-dimensional Gaussian distribution.

B. Connectivity

We show the results for both *local connectivity* and *global connectivity*. Global connectivity refers to the ratio of the number of nodes in the largest isolated component in the final key-space-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. Both global connectivity and local connectivity are affected by the key pre-distribution scheme.

1) Local Connectivity: In this experiment, we evaluate how much the deployment knowledge can improve the local connectivity. We conduct two evaluations, one for the EG scheme (i.e., $\lambda = 0$), and the other for the DDHV scheme (we set $\lambda = 19$).

A number of parameters can affect the local connectivity; to simplify the evaluation, we set $a = 0.15$ and $b = 0.10$. In addition, we make the local connectivity for $m = 100$ the same for both EG and DDHV schemes. Once these parameters are fixed, we can decide the size of the global key-space pool



(a) Comparison of EG and EG-D schemes ($\lambda = 0$). $|S| = 100,000$ and $|S_c| = 1770$.

(b) Comparison of DDHV and DDHV-D schemes ($\lambda = 19$). $|S| = 241$ and $|S_c| = 4$.

Fig. 5. Local Connectivity Performance Comparisons

and the local key-space pools. Then based on Equation (4), we can compute p_{local} for the EG, EG-D [6], DDHV, and DDHV-D schemes for various memory usage scenarios. The results are plotted in Figure 5. Figure 5(a) and Figure 5(b) clearly show that the deployment-knowledge-based EG-D and DDHV-D schemes significantly improve the local connectivity of their counterparts.

There are two "abnormal" phenomena in Figure 5(b). First, it seems that p_{local} for DDHV-D can never reach 1. The reason for this phenomenon is that some neighboring nodes might come from non-neighboring deployment groups. According to our key pre-distribution scheme, they do not share any key space because their deployment groups are not neighbors. Therefore, the local connectivity can never reach 1. The second abnormal phenomena is the discrete steps for both DDHV and DDHV-D schemes. This is because of roundings: when $|S|$ is fixed, the only parameter that can affect the local connectivity is τ, the number of key spaces carried by each sensor. Because $\tau = \lfloor \frac{m}{\lambda+1} \rfloor = \lfloor \frac{m}{20} \rfloor$, there will be discrete steps for τ when m is increased, causing the discrete steps for p_{local}.

2) Global Connectivity: It is possible that the key-space-sharing graph in our scheme has a high local connectivity, but the graph can still have isolated components. Since those components are disconnected, no secure links can be established among them. Therefore, it is important to determine whether the graph will have too many isolated components. To this end, we measure the global connectivity of the key-space-sharing graph, 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.[7]

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 [26], just like what has been done in [9], [10]. However, since neither our node distribution nor our key sharing is uniform, Erdős random graph theorem will not be a good estimation method. Recently, Shakkottai et al. have determined

[6]EG-D stands for the scheme that combines the original EG scheme and deployment knowledge. It is a special case of DDHV-D (i.e., the $\lambda = 0$ case).

[7]Some of the "unreachable" nodes 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.

TABLE I

LOCAL CONNECTIVITY VS. GLOBAL CONNECTIVITY

Local Connectivity	0.024	0.383	0.697	0.871	0.892	0.929	0.956
Global Connectivity	0.0132	0.9963	0.9988	0.9997	0.9999	0.9999	1.0000

the connectivity of a wireless sensor grid network with unreliable nodes [27]. Their results have been corrected and further improved in [28]. In our future work, we will estimate the global connectivity by using the results given in [28]. In this work, we only use simulation to estimate global connectivity. We use the configuration described in Section VII-A to conduct the simulation. The relationships between the local connectivity and the global connectivity are shown in Table I.

The simulation results indicate that when the local connectivity p_{local} reaches 0.697, only 0.12% of the sensor nodes will be wasted due to the lack of secure links; when p_{local} reaches 0.956, no nodes are wasted. These results exclude those nodes that are not within the communication range of the largest isolated component because they are caused by the deployment, not by our key pre-distribution scheme.

C. Resilience Against Node Capture

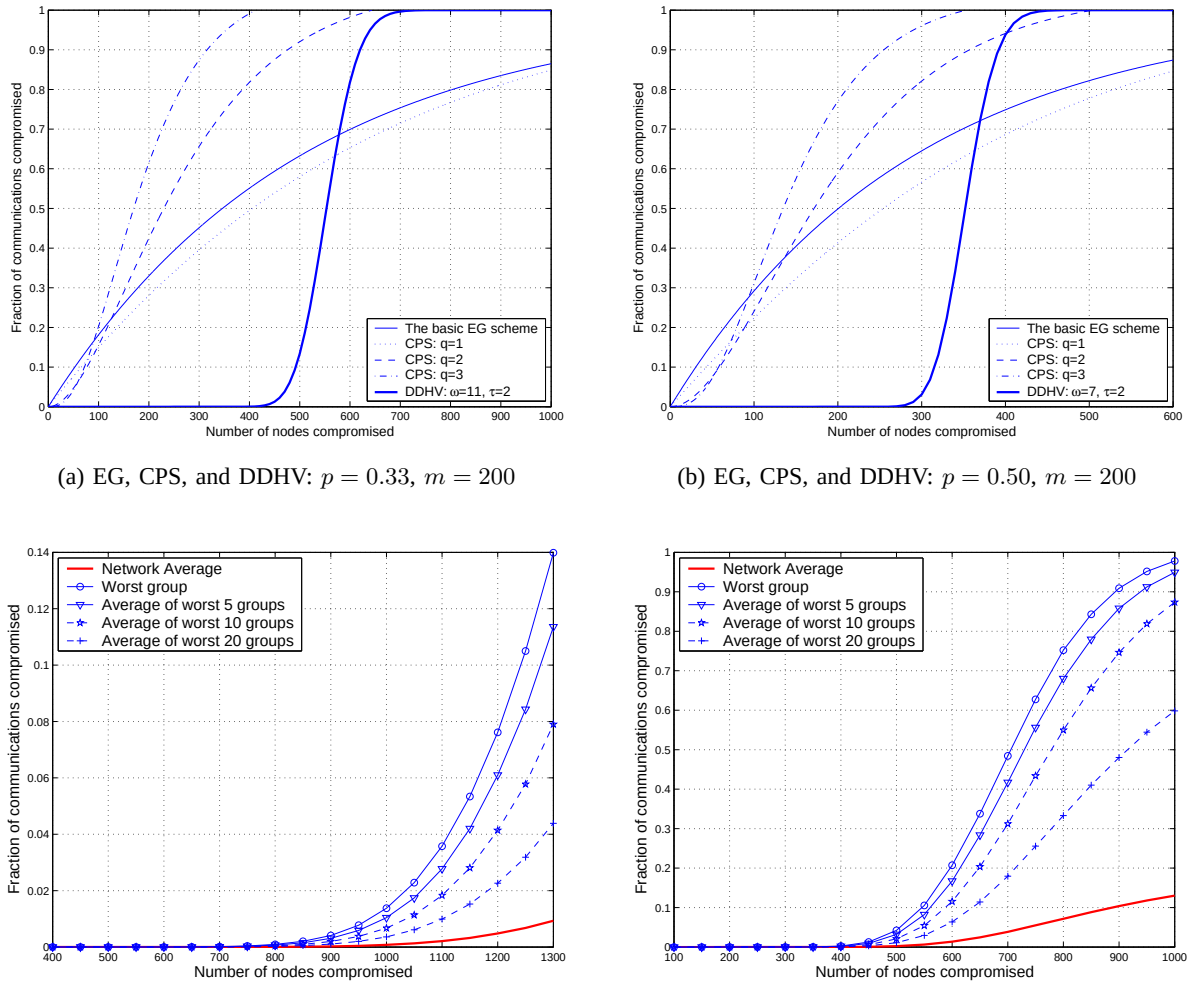
We assume that an adversary can mount a physical attack on a sensor node after it is deployed and read secret information from its memory. We need to find how a successful attack on x sensor nodes by an adversary affects the rest of the network. In particular, we want to find the fraction of additional communication (i.e., communications among uncaptured nodes) that an adversary can compromise based on the information retrieved from the x captured nodes.

1) Comparison with the Existing Schemes: In Figure 6, we show the numerical results on the resilience performance of the DDHV-D scheme against node compromise (capture). The attack circle is assumed to be $R_c = 250$ m. Our main performance metric is, P_c, the fraction of communication links that are compromised when x nodes are captured. We plot P_c for the Eschenauer-Gligor scheme (EG) [9], the Chan-Perrig-Song (CPS) scheme [10], and the DDHV scheme [11] in Figures 6(a) and 6(b). We plot P_c of the DDHV-D scheme in Figures. 6(c) and 6(d).

In Figures. 6(c) and 6(d), the network average curve shows the average of all groups in the network. Since the adversary only captures nodes inside the attack circle, only the keys of a few groups are affected. Those groups that are far away from this region are not likely to be affected at all. Therefore, the resilience performance of the network on an average is very good for the x values that we show. However, if we calculate the average resilience performance of those groups that have been affected the most, "worst groups", their resilience is quite different from the network average. For example, if we consider the worst group, P_c approaches 1 more quickly than the others. As we increase the number k in the "k worst groups" performance, P_c increases more slowly. Such a trend is shown in both of Figures. 6(c) and 6(d).

As we mentioned before, when $\lambda = 0$, the DDHV-D scheme reduces to the EG-D scheme. To see the difference between the EG scheme and the EG-D scheme, we plot the resilience of the EG-D scheme in Figure 7 for p_{local} equal to 0.33 and 0.50. Comparing Figure 7 with Figure 6, we can see that the EG-D scheme out-performs the EG scheme in resilience. However, we notice that the EG-D scheme is worse than the DDHV scheme and DDHV-D scheme. This is due to the λ value used in EG-D ($\lambda = 0$).

2) Relationships Between Resilience and Various Parameters: In the following experiments, we study how various parameters, such as memory usage m, local connectivity p_{local}, and attack radius R_c affect the resilience. For the sake of simplicity, it is better to use one value to represent the resilience, rather than using a series of values (a curve) based on x. The representative number we choose is the minimum



(a) EG, CPS, and DDHV: $p = 0.33$, $m = 200$

(b) EG, CPS, and DDHV: $p = 0.50$, $m = 200$

(c) DDHV-D: $p = 0.33$, $m = 200$, $\lambda = 19$

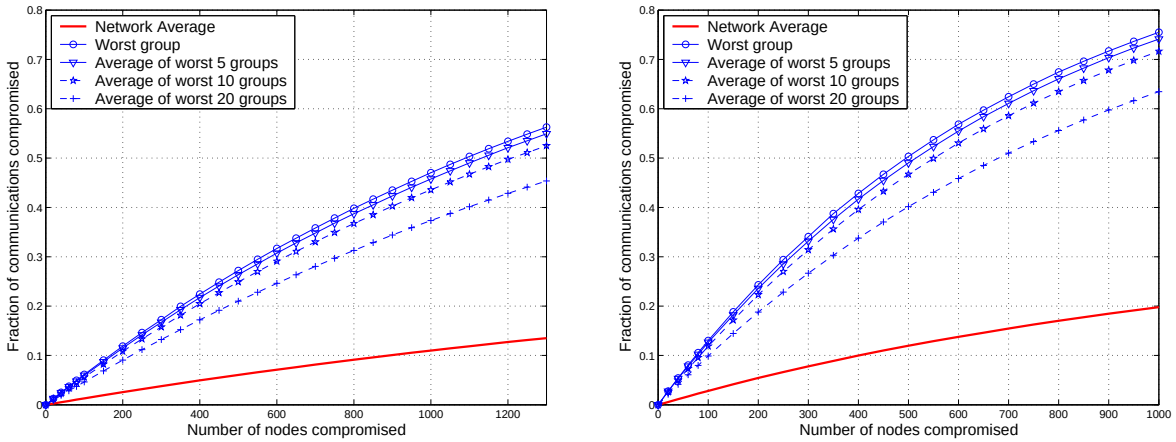
(d) DDHV-D: $p = 0.50$, $m = 200$, $\lambda = 19$

Fig. 6. Comparing the Network Resilience of EG, CPS, DDHV and DDHV-D Schemes.

number of nodes (denoted as x_{min}) that need to be compromised if attackers want to compromise at least 10% of the communication links from the worst $k = 5$ groups (excluding the ones that are connected to the compromised nodes). The reason that we choose 10% is that usually resilience deteriorates exponentially after this threshold. In the following experiments, we will use x_{min} as our resilience score and plot it on the Y-axis.

a) Resilience versus Memory Usage.: When the memory usage m increases, the local connectivity also increases. In other words, if we want to maintain the same local connectivity, we can increase the size of the global key-space pool S, such that there are more key spaces to choose from. As a result, the resilience gets better. In this experiment, we study how the increase of m affects the resilience. We fix $\lambda = 9$ and $p_{local} = 0.50$. [8] Figure 8(a) shows that resilience increases almost linearly with the memory usage.

[8]It is impossible to achieve the exact value 0.50 for the local connectivity; we maintain the value of p_{local} around 0.50.



(a) EG-D: $p = 0.33$, $m = 200$, $\lambda = 0$

(b) EG-D: $p = 0.50$, $m = 200$, $\lambda = 0$

Fig. 7. Network Resilience of the EG-D scheme (a special case of DDHV-D).

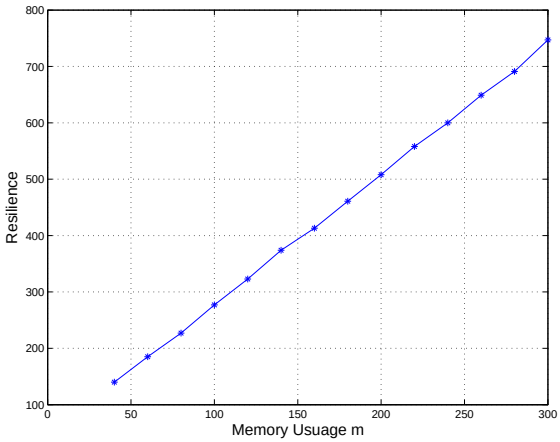
b) Resilience versus Local Connectivity.: In this experiment, we would like to answer the following question: *is it possible to achieve both high local connectivity and resilience when λ and m are fixed?* To this end, we fix $\lambda = 9$ and $m = 200$; we then change S, so p_{local} can change accordingly. We plot the resilience result for each p_{local} value. Figure 8(b) depicts the results. It clearly shows that resilience and connectivity are two conflicting properties; higher connectivity leads to lower resilience.

c) Resilience versus R_c.: The resilience of our scheme is also affected by the attack radius R_c. When the compromised nodes are more concentrated (i.e. R_c is smaller), the damage to the worst $k = 5$ groups should be more severe. To verify this hypothesis, we fix $\lambda = 9$, $m = 200$, and $p = 0.50$; we then plot the resilience results for a number of different values of attack radius R_c. Figure 8(c) depicts the results. It does show that resilience gets better when the compromised nodes are less concentrated. This result is easy to understand: when R_c gets larger, the compromised nodes become more and more evenly distributed among all the deployment groups. Therefore, given the same x (the number of compromised nodes), the number of compromised nodes for each particular deployment group is less for a larger R_c than that for a smaller R_c; thus the damage to any particular deployment group becomes less severe.

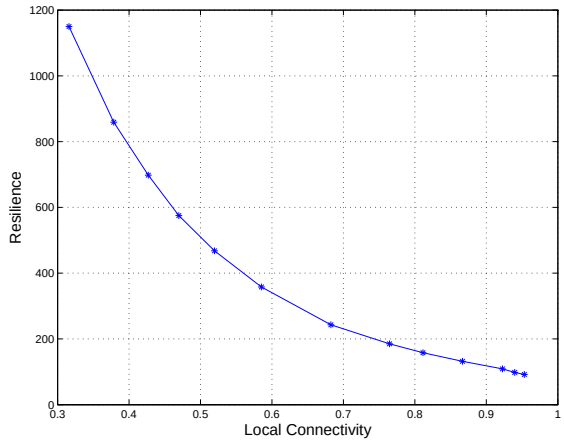
D. Communication Overhead

Since the probability that two neighboring nodes share a key space is less than one, when the two neighboring nodes do not have a common key space (i.e., they are not connected directly in the key-space-sharing graph), they need to find a route in the key-space-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}.

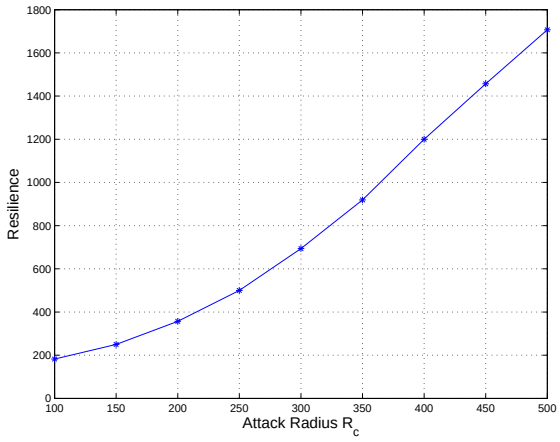
The communication overhead only depends on the local connectivity; therefore, we study the relationship between the local connectivity and the communication overhead. We use simulations to estimate how many of the key setups have to go through ℓ hops, for $\ell = 1, 2, \ldots$. Figure 9(a) depicts the communication overhead when the local connectivity changes. In Figure 9(b), we show the change of



(a) Resilience versus Memory Usage m



(b) Resilience versus Local Connectivity p_{local}



(c) Resilience versus Attack Radius R_c

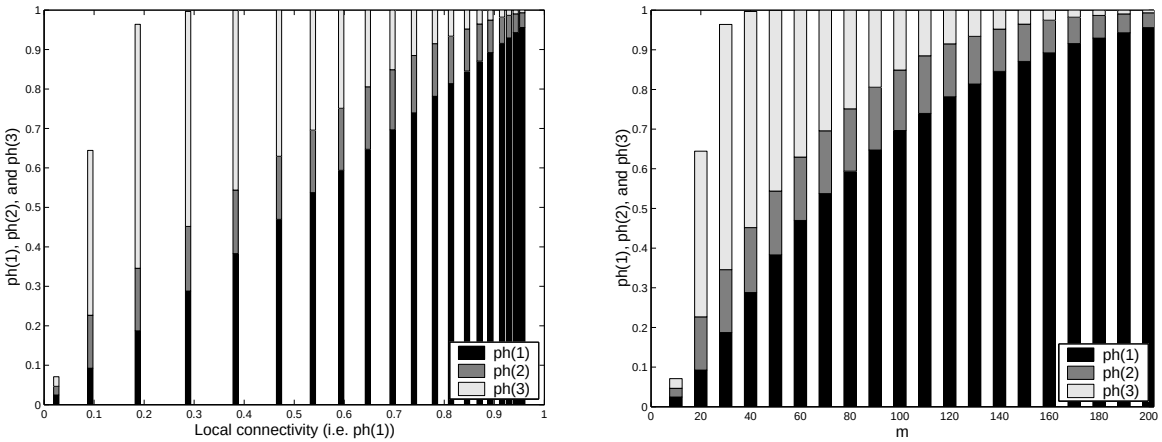
Fig. 8. Resilience as a function of Various Parameters.

communication overhead vs. memory usage m for the EG-D scheme. As we can observe from the figure, when $p_{local} = 0.3$, 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.

E. Saving on Computational Costs

Compared to the DDHV scheme, the computation for computing pairwise keys can be more efficient for the DDHV-D scheme, and can thus save energy. We explain the cause of such a difference.

According to [11], the matrix G is defined over a finite field $GF(q)$. A natural choice is to work with fields of characteristic 2 (i.e., fields of the form $GF(2^k)$) both because multiplications in this field are rather efficient and also because elements in such fields naturally map to bit strings which can then be used as cryptographic keys. In [11], it is observed that to derive a 64-bit key it is not necessary to work over $GF(2^k)$ with $k \geq 64$; instead, one can define the key as the concatenation of multiple "sub-keys" each of which lies in a smaller field. As an example, a 64-bit key can be composed of four 16-bit keys



(a) Communication Overhead versus Local Connectivity

(b) Communication overhead for the EG-D scheme ($\lambda = 0$)

Fig. 9. Communication Overhead

TABLE II
TIME (MS) FOR COMPUTING A 64-BIT SECRET KEY ($\lambda = 50$).

	eight 8-bit keys	**Four 16-bit keys**	**Two 32-bit keys**	**One 64-bit key**
Time (ms)	5.44	8.94	14.45	25.67

or eight 8-bit keys. The key observation is that security is not affected by working over $GF(q)$ where q is "small"; this is because the security arguments are information-theoretic and do not rely on any "cryptographic hardness" of the field $GF(q)$.

Since the number of multiplications for generating an 8-bit key is the same as that for a 16-bit key, and the cost of a multiplication in $GF(2^{16})$ is equivalent to four multiplications in $GF(2^8)$, using $GF(2^8)$ to generate a 64-bit key can reduce the total cost by half, compared to $GF(2^{16})$. However, there is a requirement on q: it must be larger than N, the number of columns of the matrix G in the DDHV scheme [11].

Recall that each column of the matrix G in the DDHV scheme corresponds to a node; therefore, the total number of nodes that can use a key space is the number of columns of G. We call this number the *capacity* of a key space. In the original DDHV scheme, each key space can be selected by any node in the network, so the capacity of a key space must be larger than the size of the network N. However, in the DDHV-D scheme, each key space can only be used by at most two deployment groups. Namely, the capacity of a key space can be $\frac{N}{50}$ (assuming that the total number of deployment groups is 100). This means that for $N = 10,000$, the original DDHV scheme has to work over $GF(2^{16})$, while the DDHV-D scheme can work over $GF(2^8)$.

We measured the actual time of computing a 64-bit key using a key space with $\lambda = 50$. The measurement was conducted on MICAz sensor nodes [29]. Table II describes the results for various underlying fields. The results show that being able to use $GF(2^8)$ can save 39% of energy compared to using $GF(2^{16})$.

VIII. Conclusions and Future Work

We have described a random key pre-distribution scheme that uses deployment knowledge. Our scheme takes advantage of the prior knowledge about deployment, and reduces the number of unnecessary key spaces carried by each sensor. We have conducted a comprehensive study on the connectivity and resilience of our scheme. The results have shown significant improvement in both the connectivity and resilience over the other existing key pre-distribution schemes [9]–[11]. We have presented both the analytical and numerical results. In our future work, we will study how the accuracy of the deployment model affects those results.

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 pre-distribution 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] H. Chan and A. Perrig, "PIKE: Peer intermediaries for key establishment in sensor networks," in *Proceedings of IEEE Infocom*, Miami, FL, USA, March 13-17 2005.

[14] R. Anderson, H. Chan, and A. Perrig, "Key infection: Smart trust for smart dust," in *Proceedings of IEEE International Conference on Network Protocols (ICNP 2004)*, 2004.

[15] 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.

[16] 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, lNCS Volume 435, Springer-verlag.

[17] 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, lNCS Volume 765, Springer-verlag.

[18] 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.

[19] 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.

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

[21] 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.

[22] 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, lNCS Volume 1796, Springer-verlag.

[23] R. Blom, "An optimal class of symmetric key generation systems," *Advances in Cryptology: Proceedings of EUROCRYPT 84 (Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, eds.), Lecture Notes in Computer Science, Springer-Verlag*, vol. 209, pp. 335–338, 1985.

[24] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, 2nd ed. Reading, MA: Addison-Wesley Publishing Company, Inc., 1994.
[25] C. E. Perkins, Ed., *Ad Hoc Networking*. Addison-Wesley, 2001.
[26] Erdős and Rényi, "On random graphs I," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.
[27] S. Shakkottai, R. Srikant, and N. Shroff, "Unreliable sensor grids: coverage, connectivity and diameter," in *Proceedings of the IEEE INFOCOM*, 2003, pp. 1073–1083.
[28] S. Kumar, T. H. Lai, and J. Balogh, "On k-coverage in a mostly sleeping sensor network," in *Proceedings of the 10th annual international conference on Mobile computing and networking*, 2004, pp. 144–158.
[29] CROSSBOW TECHNOLOY, INC., Available at http://www.xbow.com/.

APPENDIX I
COMPUTING $\Pr(A(n_i, n_j))$

In this appendix, we present our calculation of $\Pr(A(n_i, n_j))$, the probability that two sensors deployed from groups i and j are physical neighbors.

We divide the entire deployment region into many infinitesimal rectangular areas of size $dx\,dy$. Let $\theta = (x, y)$ represent a point in the region, and we use $\theta(dx, dy)$ to represent the infinitesimal rectangular area around θ. Assuming that the x-axis of the deployment region ranges from 0 to X and y-axis ranges from 0 to Y, we can use the following formula to compute the probability that n_i and n_j are neighbors:

$$\begin{aligned} &\Pr(A(n_i, n_j)) \\ = \quad & \int_{x=0}^{X} \int_{y=0}^{Y} Pr(A(n_i, n_j) \mid n_j \text{ is in } \theta(dx, dy)) \cdot \Pr(n_j \text{ is in } \theta(dx, dy)) \cdot dx\,dy. \end{aligned}$$

The probability that the node n_j (from group G_j) resides within this small rectangle area $\theta(dx, dy)$ can be computed directly using the probability density function f_R of the deployment:

$$\Pr(n_j \text{ is in } \theta(dx, dy)) = f_R(d_{j\theta} \mid n_j \in G_j) \cdot dx\,dy,$$

where $d_{j\theta}$ is the distance between θ and the deployment point of group j. Based on the two-dimensional Gaussian deployment distribution as shown in Eq. (1), we have

$$f_R(d_{j\theta} | n_j \in G_j) = \frac{1}{2\pi\sigma^2} e^{-\frac{(d_{j\theta})^2}{2\sigma^2}}. \tag{5}$$

Next, we show how $Pr(A(n_i, n_j) \mid n_j \text{ is in } \theta(dx, dy))$ can be computed. We use z to represent the distance from point θ to the deployment point of group G_i. We draw two circles. The first circle has a radius ℓ, and is centered at i, the deployment point of group G_i. We call this circle the i-circle. The second circle has a radius R (where R is the wireless transmission range), and is centered at $\theta = (x, y)$. We call this circle the θ-circle. When two circles intersect, we call the i-circle's arc within the θ-circle the L_{arc}, and we use $L_{arc}(\ell, z, R)$ to represent the length of the arc. We now consider an infinitesimal ring area $L_{arc}(\ell, z, R) \cdot d\ell$. The bold areas in Figure 10(a) and 10(b) show the infinitesimal ring areas. Using geometry, we can compute the length of the arc using the following formula:

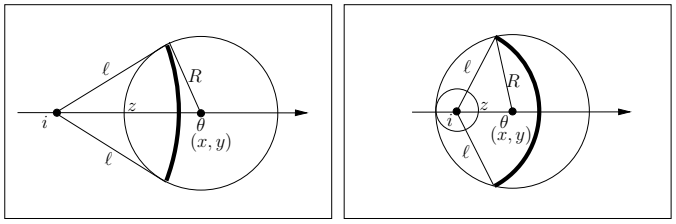
$$L_{arc}(\ell, z, R) = 2\ell \cos^{-1}\left(\frac{\ell^2 + z^2 - R^2}{2\ell z}\right).$$

Recall that $f_R(\ell \mid n_i \in G_i)$ represents the probability density function of the deployment for group G_i. Therefore, the probability that the node n_i resides within this small ring area is

$$f_R(\ell \mid n_i \in G_i) \cdot L_{arc}(\ell, z, R) \cdot d\ell.$$

We define $g(z \mid n_i \in G_i)$ as the probability that the sensor node n_i from group G_i resides within the θ-circle, where z is the distance between θ and the deployment point of group G_i. It is not hard to see that $Pr(A(n_i, n_j) \mid n_j \text{ is in } \theta(dx, dy)) = g(z \mid n_i \in G_i)$.

To calculate $g(z \mid n_i \in G_i)$, we integrate the probabilities over all the ring areas (for different ℓ) within the θ-circle. Therefore, when $z > R$ (as shown in Figure 10(a)),



(a) i is outside of the circle ($z > R$) (b) i is inside of the circle ($z < R$)

Fig. 10. Probability of nodes residing within a circle.

$$g(z \mid n_i \in G_i) = \int_{z-R}^{z+R} f_R(\ell \mid n_i \in G_i) \cdot L_{arc}(\ell, z, R)\, d\ell \ . \tag{6}$$

When $z < R$ (as shown in Figure 10(b)),

$$g(z \mid n_i \in G_i) = \int_{0}^{R-z} \ell \cdot 2\pi f_R(\ell \mid n_i \in G_i)\, d\ell + \int_{R-z}^{z+R} f_R(\ell \mid n_i \in G_i) \cdot L_{arc}(\ell, z, R)\, d\ell \ . \tag{7}$$

Combining the above equations together, we get

$$Pr(A(n_i, n_j)) \quad = \quad \int_{y=0}^{Y} \int_{x=0}^{X} f_R(d_{j\theta} \mid v \in G_j) \cdot g(d_{i\theta} \mid u \in G_i) \cdot dx\, dy, \tag{8}$$

where $d_{i\theta}$ (resp. $d_{j\theta}$) is the distance between the deployment point of G_i (resp. G_j) and $\theta = (x, y)$.

Appendix II
Computing $\Pr(B(n_i, n_j))$

In this appendix, we calculate $\Pr(B(n_i, n_j))$, the probability that two sensors deployed from groups i and j share at least one common key. The probability of this event does not depend on the deployment knowledge. It only depends on the key pre-distribution, i.e., the key-space pools, shared key spaces between key-space pools, and the number of key spaces each sensor carries.

Let $\xi(i, j)$ represent the number of shared key spaces between the deployment groups G_i and G_j. According to our key-space pool construction scheme, we have the following:

$$\xi(i,j) = \begin{cases} |S_c|, & \text{when } i = j; \\ \xi_a = \lfloor a|S_c| \rfloor, & \text{when } i \text{ and } j \text{ are horizontal or vertical neighbors;} \\ \xi_b = \lfloor b|S_c| \rfloor, & \text{when } i \text{ and } j \text{ are diagonal neighbors;} \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

To calculate $\Pr(\text{two nodes do not share any key space})$, we use the following strategy: the first node selects k key spaces from the ξ shared key spaces, it then selects the remaining $\tau - k$ key spaces from the non-shared key spaces. To avoid sharing any key space with the first node, the second node cannot select any of the k key spaces from those ξ shared key spaces that are already selected by the first node, so it has to select τ key spaces from the remaining $(|S_c| - k)$ key spaces from its key-space pool. Therefore, $p(\xi(i, j))$, the probability that two nodes share at least one key space when their key-space pools have $\xi(i, j)$ key spaces in common, can be calculated in the following:[9]

[9]When $\xi(i, j) = |S_c|$, $p(\xi(i, j))$ can be simplified to $1 - \frac{\binom{|S_c|-\tau}{\tau}}{\binom{|S_c|}{\tau}}$; when $\xi(i, j) = 0$, $p(\xi(i, j)) = 0$.

$$\begin{aligned} & \Pr(B(n_i,n_j)) = p(\xi(i,j)) = 1 - \Pr(\text{two nodes do not share any key space}) \\ = \; & 1 - \frac{\sum_{k=0}^{min(\tau,\xi(i,j))} \binom{\xi(i,j)}{k}\binom{|S_c|-\xi(i,j)}{\tau-k}\binom{|S_c|-k}{\tau}}{\binom{|S_c|}{\tau}^2} \;, \end{aligned} \tag{10}$$

where $\xi(i,j)$ is given by Equation (9).

APPENDIX III
RESILIENCE ANALYSIS

We present our detailed derivation of $\Pr(c \text{ is compromised} \mid C(x,y) \text{ and } A(u,v) \text{ and } B(u,v))$ of Section VI-B in this appendix.

Let K_i be the event that c is using a key space associated with group i. Then

$$\begin{aligned} & \Pr(c \text{ is compromised} \mid C(x,y) \text{ and } A(u,v) \text{ and } B(u,v)) \\ = \; & \sum_{i\in\Psi} \Pr(c \text{ is compromised} \mid K_i \text{ and } C(x,y) \text{ and } A(u,v) \text{ and } B(u,v)) \cdot \\ & \Pr(K_i \mid A(u,v) \text{ and } B(u,v)) \;. \end{aligned} \tag{11}$$

The last equation is obtained due to the fact that K_i is independent to $C(x,y)$.

According to the result given in [11], for any of the $|S_c|$ keys belonged to group i that might be used by any link, we have

$$\begin{aligned} & \Pr(c \text{ is compromised} \mid K_i \text{ and } C(x,y) \text{ and } A(u,v) \text{ and } B(u,v)) \\ = \; & \sum_{j=\lambda+1}^{X_i(x,y,R_c)} \binom{X_i(x,y,R_c)}{j} \left(\frac{\tau}{|S_c|}\right)^j \left(1-\frac{\tau}{|S_c|}\right)^{X_i(x,y,R_c)-j} \;. \end{aligned}$$

Now we need to calculate the probability

$$\Pr(K_i \mid A(u,v) \text{ and } B(u,v)) = \frac{\Pr((K_i \text{ and } B(u,v)) \text{ and } A(u,v))}{\Pr(A(u,v) \text{ and } B(u,v))} \;,$$

in (11). Note that, the probability $\Pr(A(u,v) \text{ and } B(u,v))$ has been given in the previous subsection. Since that the event K_i is true implies that the event $B(u,v)$ is true, we get

$$\Pr((K_i \text{ and } B(u,v)) \text{ and } A(u,v)) = \Pr(K_i \text{ and } A(u,v)).$$

By a similar procedure given in previous subsection we have

$$\Pr(K_i \text{ and } A(u,v)) = \frac{1}{(nt)^2} \sum_{j\in\Psi_i} p(\xi(i,j)) \cdot Pr(A(u,v) \mid u \in G_i \text{ and } v \in G_j).$$

Combing the above Equations and Equation (10), we have

$$
\begin{aligned}
& \Pr(c \text{ is compromised} \mid A(u,v) \text{ and } B(u,v)) \\
= \; & \frac{1}{XY}\int_{y=0}^{Y}\int_{x=0}^{X} \Pr(c \text{ is compromised} \mid C(x,y) \text{ and } A(u,v) \text{ and } B(u,v))\; dxdy \\
= \; & \frac{1}{XY}\int_{y=0}^{Y}\int_{x=0}^{X} \sum_{i\in\Psi}\left\{ \sum_{j=\lambda+1}^{X_i(x,y,R_c)} \binom{X_i(x,y,R_c)}{j} \left(\frac{\tau}{|S_c|}\right)^{j} \left(1-\frac{\tau}{|S_c|}\right)^{X_i(x,y,R_c)-j} \right. \\
& \left. \cdot \frac{\sum_{j\in\Psi_i} p(\xi(i,j))\cdot Pr(A(n_i,n_j))}{\sum_{j\in\Psi}\sum_{i'\in\Psi} p(\xi(i',j))\cdot Pr(A(n_{i'},n_j)} \right\}\; dxdy, \qquad (12) \\
= \; & \frac{1}{XY}\cdot \sum_{i\in\Psi} \frac{\sum_{j\in\Psi_i} p(\xi(i,j))\cdot Pr(A(n_i,n_j))}{\sum_{j\in\Psi}\sum_{i'\in\Psi} p(\xi(i',j))\cdot Pr(A(n_{i'},n_j)} \cdot \\
& \int_{y=0}^{Y}\int_{x=0}^{X} \left\{ \sum_{j=\lambda+1}^{X_i(x,y,R_c)} \binom{X_i(x,y,R_c)}{j} \left(\frac{\tau}{|S_c|}\right)^{j} \left(1-\frac{\tau}{|S_c|}\right)^{X_i(x,y,R_c)-j} \right\}\; dxdy, \qquad (13)
\end{aligned}
$$

where $Pr(A(n_i,n_j))$ is given in Equation (8).