Key Predistribution Schemes for Sensor Networks for Continuous Deployment Scenario[*]

Abdülhakim Ünlü, Önsel Armagan, Albert Levi, Erkay Savas, and Özgür Erçetin

Sabanci University, Istanbul, Turkey
{aunlu, onsel}@su.sabanciuniv.edu
{levi, erkays, oercetin}@sabanciuniv.edu

Abstract. In sensor networks, secure communication among sensor nodes requires secure links and consequently secure key establishment. Due to resource constraints, achieving such key establishment is non-trivial. Recently some random key predistribution techniques have been proposed to establish pairwise keys. Some of these approaches assume certain deployment knowledge is available prior to deployment and nodes are deployed in groups/bundles. In this paper, we propose another practical deployment model where nodes are deployed over a line one by one in a continuous fashion. In this model, sensor nodes can also be deployed over multiple parallel lines to cover two-dimensional area. Based on this model, we develop two key predistribution schemes. Analysis and simulation results show that our key predistribution schemes make use of the deployment knowledge better than the existing schemes. Thus they perform better than other location-aware protocols using the metrics of connectivity, resiliency, memory usage and communication cost for key establishment.

1 Introduction

In sensor networks [1], confidentiality, privacy and authenticity of communication between sensor nodes are important when nodes are deployed in an environment where there are adversaries. In order to fulfill these security requirements, cryptographic techniques are employed. Generally symmetric cryptography is used to provide security in sensor networks. In order to use symmetric key cryptography, communicating sensor nodes must share the same key. Distribution of keys to large amount of sensor nodes, so that they can establish secure links, is an active research area. Generally *key predistribution schemes* [2-8], where the keys are stored in sensor nodes before deployment, are used for this purpose.

A naïve way of key predistribution is to generate a master key and install this master key to all nodes before the deployment. However in this scheme, when a node is captured, the master key is also captured and all secure links in the sensor network are compromised.

[*] Albert Levi and Abdülhakim Ünlü are supported by the Scientific and Technological Research Council of Turkey under project number 104E071. Erkay Savas is supported by the Scientific and Technological Research Council of Turkey under project number 104E007.

Another extreme key predistribution way is to assign unique link keys for each node. In this method, compromise of one node leads to compromise of only that node's links. However, this method is not scalable since the total number keys to be predistributed per node should be as much as the number of nodes in the network in order to guarantee that after deployment each neighboring node pair shares a key.

In order to overcome this scalability problem and effectively use the node memory, Eschenauer and Gligor proposed a probabilistic key predistribution scheme [5]. In this scheme, before sensor deployment, a key server creates a key ring for each node, by picking a limited amount of random keys from a large key pool. Then the key server loads the key ring to memory of each node. After deployment, sensor nodes in the field let their neighbors know which keys they have. If two neighboring nodes share one or more identical keys, then they can establish a secure link. After this shared key discovery with direct neighbors, neighboring node pairs that do not share keys can establish secure links in multiple hops. If the local connectivity (in terms of secure links) is above a certain threshold, then random graph theory [9] states that overall sensor network will be cryptographically connected with high probability. Du et al. utilized Blom's key management scheme [12] in a key predistribution scheme for sensor networks [4]. This scheme shows a threshold property; until ? nodes are captured, the network is perfectly secure, but after ? nodes are compromised all secure links are compromised.

Some recent papers on random key predistribution [3,7,10,11] utilized expected location information of sensor nodes in their models. In all these *location-aware* approaches, it is assumed that nodes are prepared in small groups and deployed as bundles, e.g. groups of nodes can be dropped from a plane, similar to parachuting troops or dropping cargo. The nodes in the same group have a very large chance to be in the range of each other. Moreover, the node groups that are dropped next to each other also have a chance to be close to each other on the ground. Using this deployment location knowledge, key pools and key rings are arranged and performance of key predistribution schemes can be improved substantially. In location aware schemes, the node deployment model is one of the most important design criteria that directly affects the performance of the scheme. As discussed above, a batch deployment strategy is assumed in the location aware random key predistribution schemes proposed in the literature. Such a deployment strategy may not be appropriate for scenarios like borderline or perimeter defense - if sensors are deployed in bundles, it is likely that there will be places on the border with a few or no sensor nodes. Moreover, there is still room to further improve the performance, in terms of connectivity, resiliency and memory usage, of location-aware key predistribution schemes with more realistic deployment models.

1.1 Our Contribution

We introduce a new deployment model, called the *continuous deployment model*, and develop two key pre-distribution schemes on this model. The main idea behind the continuous deployment model is to drop the nodes one by one (i.e. not in batches) continuously from an aerial vehicle. The aerial vehicle may follow a continuous line for perimeter defense applications. In applications that need area coverage, the vehicle may follow a route with several parallel lines. We use the latter scenario, which is more complicated than the former one, in the development and the analysis of key pre-distribution schemes developed based on the continuous deployment model. In our first key pre-distribution scheme, it is assumed we know the order in which the nodes are dropped off for each line. For key predistribution in this scheme, we take a deterministic approach and assign pairwise keys to sensor nodes. In our second key predistribution scheme, we relaxed the order assumption such that the dropping order of the sensor nodes is not known, but the nodes to be dropped for each line are grouped. Here we use a probabilistic key predistribution mechanism; for each line, each node is assigned some keys from the key pools.

We anticipate that the use of more deployment knowledge, as in the methods that we proposed, would improve the performance of the system. We performed analytical and simulation-based performance evaluation of the proposed schemes and show that the proposed approach actually improves key predistribution performance over Du et al.'s scheme [3], in which the nodes are deployed in groups, in terms of connectivity, resiliency against node capture and memory usage.

2 The Continuous Deployment Model

In this section, we introduce a practical deployment model, where nodes are deployed sequentially but not in batches. In our deployment model the nodes are dropped one by one following a trajectory. This model can be easily realized by dropping nodes through a pipe in a plane as the plane flies over a known route. For example, if a rectangular area is to be covered with sensor nodes, the plane takes a route where it scans the rectangular area line by line. Figure 1 shows an example sensor network deployed in this model.

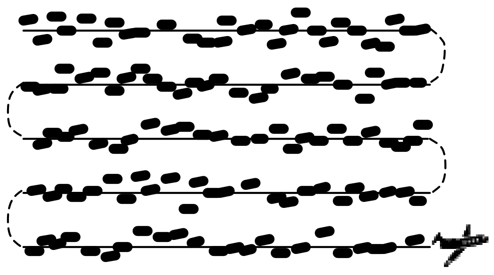

Fig. 1. A sample sensor network

The point where a node is dropped out of plane or helicopter is called its *deployment point*. However, due to several reasons its actual position drifts from deployment point. The actual position of a node on the field after deployment is named its *resident point*. Both deployment and resident points are defined in two-dimensional space. In the rest of the paper, the deployment area is assumed to be a rectangular one. In this area, there are L parallel lines and N nodes per line. Our deployment model assumes fixed intervals between the deployment points of two consecutive nodes of a line. The deployment point of i^{th} node on j^{th} line is denoted as d_{ji}, where j=1 ... L and i=1 ... N. Similarly the resident point of that node is denoted as r_{ji}.

The resident point of a node may float away from its deployment point. Due to this fact, two nodes with deployment point d_{li} and d_{lj}, where $d_{li} < d_{lj}$, can be at resident points r_{li} and r_{lj} where $r_{li} > r_{lj}$. In our model, two nodes can be neighbors according to their deployment points, but they can be out of each others' coverage after deployment. We call two nodes *neighbors* only if their *resident points* are close enough so that they can directly communicate over radio. The density of the lines and node dropping frequency are important system parameters to keep the resulting sensor network connected. Our model utilizes two-dimensional Gaussian distribution function to determine probability of a node being at a resident point based on its deployment point.

3 Continuous Key Predistribution Scheme

Key distribution for our deployment model can be performed in two ways.

In the first way, we assume that the deployment order of individual nodes is known. In this way, the neighboring relationships, in terms of the deployment points, are known. Such knowledge yields very efficient key distribution method that will be discussed later. However, in order to realize this, we have to transfer cryptographic materials to the nodes just before dropping them, so we need to have a complex setup inside the plane. Alternatively, we may transfer cryptographic materials before loading them to plane, but we have to preserve nodes' order by, for example, keeping all the sensor nodes in pipes.

In the second way, a line of nodes is treated as a single group. We do not assume knowledge of order of nodes; we just form groups of nodes and then store cryptographic material according to the key distribution scheme that will be explained later. Then, we deploy each group as a line in a random order. This approach is simpler to realize than the first method, but it has some performance deficiencies that will be discussed in this paper.

We propose two different key predistribution schemes, Scheme I and Scheme II, for the above two ways. Both of them follow well-known three phase approach as in other key predistribution schemes proposed in the literature. First phase is the "predistribution" phase, where keys are stored in nodes according to a method proposed by the scheme. Second phase is the "direct key establishment" phase, where nodes discover their neighbors and find out if they share common keys with their neighbors to form secure link. Third phase is the "path key establishment phase", where a node tries to find secure paths to its neighbors, with which it does not share common keys, in order to establish secure link. A *secure link* exists between two nodes if they both own at least one key in common and they are neighbors. We assume that all keys have unique IDs.

3.1 Key Predistribution Scheme I

Parameters and the symbols used in this scheme are:

N	number of nodes on a line
L	number of lines that makes up the sensor network
M	number of keys shared with nodes on the same line
Q	number of keys shared with nodes on adjacent line
d	distance between deployment points on a line

A	radio range of a node
L_i	ith line, where i=1 .. L
d_{ij}	deployment point of jth node on line L_i, j=1 .. N
s_{ij}	the id of the sensor node with deployment point d_{ij}
r_{ij}	resident point of jth node on line L_i, where j=1 .. N

Along with the deployment model examined in the previous section, sensor nodes, which are placed adjacent in the pipe, have high probability of being neighbors after deployment. Similarly, sensor nodes, which have similar locations in consecutive pipes, maintain the likelihood of being neighbors. As a result of this observation, we infer that a pairwise key predistribution method would work efficiently. Thus, we adopted such a strategy in our method. There are three phases in this scheme as described above: (i) Predistribution Phase, (ii) Direct Key Establishment Phase, and (iii) Path Key Establishment Phase.

Predistribution phase. This phase is split into two: *inline key predistribution* and *cross-line key predistribution*. Inline key predistribution is for the nodes within the same line of deployment. Cross-line key predistribution is for the nodes in adjacent lines. Figure 2 depicts this phase.

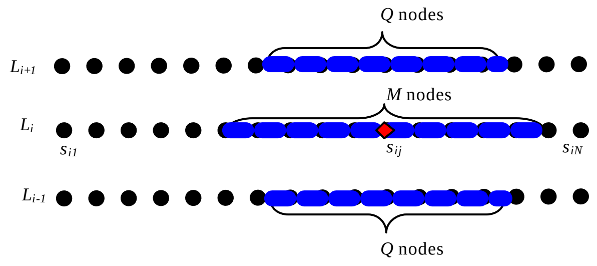

Fig. 2. Node s_{ij} shares keys with square-shaped nodes

Inline Key Predistribution. The setup server creates and stores pairwise keys in sensor nodes such that each node shares keys with its M neighbors on the current line. More formally, for all i=1 .. L, and j=1 .. N, the setup server creates M keys to be stored in s_{ij} such that s_{ij} and its M neighboring sensor nodes, $s_{i(j-M/2)},\ldots,s_{i(j-1)},s_{i(j+1)},\ldots,s_{i(j+M/2)}$,share unique keys.

Cross-Line Key Predistribution. Sensor nodes also share keys with their neighbors in neighboring lines. For all i=1 .. L, and j=1 .. N, the setup server creates 2*Q keys to be stored in s_{ij} such that this node shares unique pairwise keys with Q nodes from the lower line, $s_{(i-1)(j-Q/2)},\ldots\ldots,s_{(i-1)(j+Q/2)}$, and also Q nodes from an upper line, $s_{(i+1)(j-Q/2)},\ldots\ldots,s_{(i+1)(j+Q/2)}$.

After these two processes, a sensor node will have $M+2Q$ keys before deployment.

Direct Key Establishment Phase. After deployment, sensor nodes communicate with their neighbor nodes to discover shared keys in order to establish secure links Shared key discovery is trivial, since sensor nodes already have IDs of nodes with which they share pairwise keys. So a node only needs to know the IDs of its neighbors. When a node finds a matching node in its neighborhood, they can immediately start using their pairwise shared key. Unauthorized entities cannot know the IDs of keys used to secure links or IDs of keys in any node, since only node IDs are to be transmitted over unencrypted links,. This phase is indifferent for both nodes on the same line and nodes on the adjacent lines.

Path Key Establishment Phase. After direct key establishment phase, a node may end up with a case where it has neighbors that it cannot find a shared key to establish a secure link. Thus, these two neighboring nodes without a secure link will have to find a secure path, which is a path of secure links, through their other neighbors. The process of establishing a secure link over a secure path is called *path key establishment*. The process works as follows. Assume node s_{ij} does not have a secure link with its neighbor node s_{ik}. Node s_{ij} asks its 1-hop neighbors, with which it has secure links, to see if they also have secure links with node s_{ik}. If any of the neighbors, say s_{in}, has such a secure link, then s_{in} generates a random key and sends the key to both node s_{ij} and s_{ik} over secure links. Then the nodes s_{ij} and s_{ik} use this key to establish a secure link. If none of the 1-hop neighbors have secure link with node s_{ik}, then node s_{ij} asks its 2-hop neighbors. If not found again, s_{ij} asks to next hop neighbors until it finds a node that shares key with s_{ik}. If the graph of secure links is a connected graph, a node eventually finds a secure path to any node in the sensor network.

In our analysis, we will show that a node can reach all its neighbors with high probability in three hops of secure links.

3.2 Key Predistribution Scheme II

Parameters and the symbols used in this scheme are:

N	number of nodes on a line
L	number of lines that makes up the sensor network
L_i	i^{th} line, where i=1 .. L
S_i	key space of line i
s_I	number of nodes a key in S_i is distributed on L_i
s_c	number of nodes a key in S_i is distributed on neighbors of L_i
M_I	memory space of nodes of L_i for keys from S_i
M_c	memory space of nodes of neighbors of L_i for keys from S_i
K	number of unique keys in a key space
d	distance between deployment points on a line
A	radio range of a node
A_{ij}	circular area around node s_{ij}, where s_{ij} can send and receive radio signals
d_{li}	deployment point of node i on line l
r_{li}	resident point of node i on line l
s_{ij}	the id of sensor node with deployment point d_{ij}
k_{ij}	the id of j^{th} key in key space S_i

In this model, we do not assume a particular order in the deployment points of the nodes of a line. Thus we use less deployment knowledge as compared to predistribution scheme I. Although the scheme is still based on pairwise key distribution some redundancy should also be added in order to achieve a reasonable level of connectivity.

Setup server generates groups of unique keys for each line. This keys form the key space, S_i, of line i. There are K keys in each key space, and a node from line i gets keys from S_i, S_{i-1} and S_{i+1} according to the key predistribution method. The duplication of each key is limited and determined parametrically.

Similar to Scheme I, this scheme has three phases; predistribution, direct key establishment and path key establishment.

Predistribution Phase. In key predistribution step, we describe the method how keys are distributed to nodes on various lines. Setup server generates key spaces for each line, S_i, where $i = 1 .. L$, then distributes s_I and s_c copies of each key as explained below. Our aim here is to distribute the keys such that nodes that are expected to be near share more keys. Key predistribution method for each k_{ij}, where $i = 1 .. L$ and j = 1.. N, is as follows:

1. Key k_{ij} is randomly generated for key space of S_i of line L_i.
2. s_I nodes with sufficient space in their M_I are randomly selected on L_i and k_{ij} is installed in those s_I nodes.
3. s_c nodes with sufficient space in their M_c are selected randomly from each neighboring lines of L_i. So, $2s_c$ nodes are selected from two neighboring lines. Then k_{ij} is installed in those s_c nodes in each neighboring line.

At the end of key predistribution phase, each key from key space S_i has a total of ($s_I + 2s_c$) copies on three lines; s_I copies in L_i and $2s_c$ copies in L_{i-1} and L_{i+1}. And each node has a total of $M_I + M_c$ keys installed.

We can calculate K, the size of each key space S_i, $i = 1 .. L$, by using s_c, s_I, M_I, and M_c. Since there are N sensor nodes on line i, and since setup server loads exactly M_I unique keys from S_i into each node on line i, setup server will need NM_I keys. Each key from S_i will have s_I copies on line i. Also, s_c copies of keys from S_{i+1} and S_{i-1} will be loaded into nodes from line i, and each node has M_c memory for keys from neighboring key spaces. Then, number of unique keys in any key space, K, can be computed as follows:

$$K = NM_I / s_I = NM_c / 2s_c$$

Direct Key Establishment Phase. After deployment, nodes have to find shared keys with its neighbors. This phase is similar to the basic scheme [5]. Here, each node needs to know which keys its neighbors have so that it can decide which keys they share. Each node broadcasts a message containing the indices of the keys it carries. Nodes can use these broadcast messages to find out if they share common keys with their neighbors. If a node can find a shared key with one of its neighbors, it can use that key to establish a secure link between itself and its neighbor.

Path Key Establishment Phase. If two neighboring nodes cannot find a shared key directly, they have to reach a common key over a secure path. This method is identical to the path key establishment method in Scheme I.

4 Performance Analysis

In our analysis and simulation, we use the following configuration. Deployment area is 1000m x 1000m. There are 50 deployment lines, i.e. L=50, and the distance between lines is 20m. On each deployment line there are 200 nodes, i.e. N=200. Total number of sensor nodes, NxL, is 10000. Distance between two adjacent deployment points, d, is 5m. Communication range, R, for each node is 40m. Standard deviation of normal distribution, s, is 10m. For scheme II, total number of unique keys is 50K = 100000.

4.1 Local and Global Connectivity

In this section, we show our simulation results of the probability of a node sharing a key with its neighbors. This probability is called *local connectivity*, P_{local}. The detailed formulation for P_{local} could not be given here due to space limitations. Figure 3 shows local connectivity versus memory usage m. We compare results for our scheme I and scheme II with Du et al's [3] scheme. Scheme II has higher connectivity than [3] for all values of m.

For scheme II, different values of s_I and s_c results in different P_{local} values even for the same memory usage. In our experiments for various m values, we obtained best results when s_I and s_c are equal.

Scheme I outperforms both scheme II and Du's scheme for low m values. In our simulations, scheme I reached a maximum local connectivity value of 0.8518 at M=28 and Q=26 that yields m=80. As the number of keys used increases after m=80, local connectivity stays the same. Increasing M and Q, and consequently m, values means that a node shares keys with distant nodes. This will not contribute to the local connectivity, because distant nodes have very small probability of falling within that node's communication range. Simulation results in Figure 3 confirm our explanation.

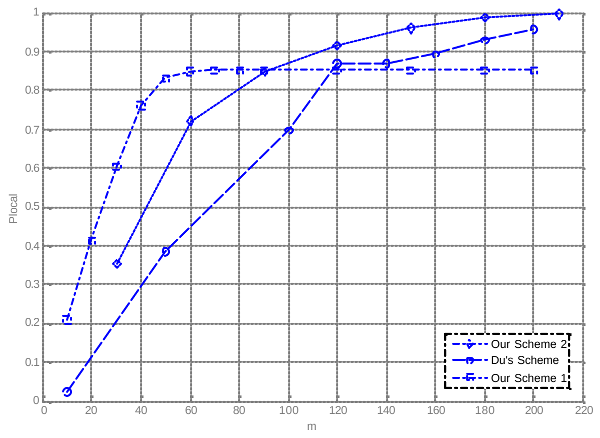

Fig. 3. Local connectivity versus memory usage *m*

There are two factors that makes scheme II's connectivity performance better than Du et al.'s scheme. Firstly, in our schemes we use more deployment knowledge such that in Du’s scheme there is a single deployment point for each bundle of nodes, whereas in scheme II there are deployment points for each node. Secondly, in scheme II, we distribute copies of a key homogeneously. We distribute copies of a key to both upper and lower neighboring lines, so a node can use keys in its M_c to establish secure links with nodes on the same line, on its direct neighbor lines and nodes on two lines away. In addition, by introducing s_I and s_c, we can have a fixed number of copies of all keys. Du’s scheme can have the same average number of copies of keys with same m values but a particular key can have a much higher or much lower number of copies. Fixing number of copies in scheme II contributes to homogeneity of key distribution.

A high local connectivity value means that a node can communicate with most of its neighbors securely. However, a high local connectivity value does not guarantee that there will not be isolated parts in the network. Thus, we need to examine that whether our schemes can create too many isolated components or not. We measured, global connectivity, which is the ratio of size of largest isolated part to the size of whole network, through simulations. The results show that 100% global connectivity is reached when m is as low as 10 for Scheme I and 30 for Scheme II.

Since we determine the deployment point of all nodes in Scheme I and fix the number of copies of a key in Scheme II, we minimize the possibility that network has more than one isolated part. Our simulation results support this idea.

4.2 Resiliency Against Node Capture

We investigate the effects of compromised nodes on direct key establishment. We assume that total c randomly chosen nodes are compromised. The fraction of additional communications that can be compromised based on the information from the compromised nodes defines the resiliency of our system. This section is focused on the resiliency of *Scheme II* against node capture attacks. *Scheme I* uses pairwise keys, therefore it is %100 resilient against compromising of sensor nodes.

Let k_{aj} denote a key generated for line L_a. Assume k_{aj} is used for a link between two nodes that are not compromised. We know that there are S_i copies of k_{aj} on the nodes of L_a, S_c copies of k_{aj} on the nodes of $L_{a\text{-}1}$ and S_c copies of k_{aj} on the nodes of L_{a+1}. Thus, in order to compromise k_{aj}, adversary should compromise nodes from $L_{a\text{-}1}$, L_a, and L_{a+1}. If there are j compromised nodes on L_a, the probability that k_{aj} is not compromised on line L_a is given as:

$$p_{comp\_i}(j) = \binom{N-j}{s_I} \Big/ \binom{N}{s_I} \tag{1}$$

If there are j compromised nodes on an adjacent line of L_a, the probability that k_{aj} is not compromised on that adjacent line is given as:

$$p_{comp\_c}(j) = \binom{N-j}{s_c} \Big/ \binom{N}{s_c} \tag{2}$$

Thus, if there are x compromised nodes on $L_{a\text{-}1}$, y compromised nodes on L_a and z compromised nodes on L_{a+1}, the probability that k_{aj} is not compromised becomes $P_{comp\_c}(x)*P_{comp\_i}(y)*P_{comp\_c}(z)$ .The probability that there are x compromised nodes on line L_{a-1}, y compromised nodes on line L_a and z compromised nodes on line L_{a+1} is calculated as:

$$p_{c\_xyz} = \binom{c}{x}\binom{c-x}{y}\binom{c-x-y}{z}\left(\frac{1}{L}\right)^{x}\left(\frac{1}{L}\right)^{y}\left(\frac{1}{L}\right)^{z}\left(\frac{L-3}{L}\right)^{c-x-y-z} \quad (3)$$

By using equations 1, 2, and 3, we calculate the probability that an adversary obtains a key, which is used for a link between two nodes that are not compromised, out of randomly compromised c nodes as given below:

$$p_{comp\_all}(c) = 1 - \sum_{x=0}^{N}\sum_{y=0}^{N-2}\sum_{z=0}^{N}\binom{c}{x}\binom{c-x}{y}\binom{c-x-y}{z}\left(\frac{1}{L}\right)^{x+y+z}\left(\frac{L-3}{L}\right)^{c-x-y-z} P_{comp\_c}(x)P_{comp\_i}(y)P_{comp\_c}(z) \quad (4)$$

Comparison of our scheme II and Du et al.'s [3] scheme is shown in Figures 4 and 5. In both schemes probability of a link being compromised, P_{comp_all}, is plotted against number of nodes captured. In Figure 4, number of keys in a node is taken as 60 and number of nodes is 10000 for both schemes. In Figure 5, we fix local connectivity to 0.86 for both schemes. Our scheme is outperforms Du's scheme, because we can reach a local connectivity of 0.86 with only m=90 keys in a node, whereas Du's scheme requires m=140 to reach the same local connectivity.

Probability of a secure link being compromised when a number of nodes are captured is directly proportional to the number of copies of a key. In scheme II, number of copies of a key is a parameter determined by s_I and s_c. In Figure 5, for scheme II there are 3+2*3 = 9 copies of a key. In Du et al.'s scheme, a key has a random number of copies but we can find an average number of copies of a key by using $|S|$, number of unique keys in the sensor network, N, number of total nodes and m, number of keys in each node: $k_{average} = N \cdot m / |S|$. Because we used the same m values for both scheme II and Du's scheme in Figure 4, there were six copies of a key for both schemes and we got very similar results for both schemes as shown in the figure.

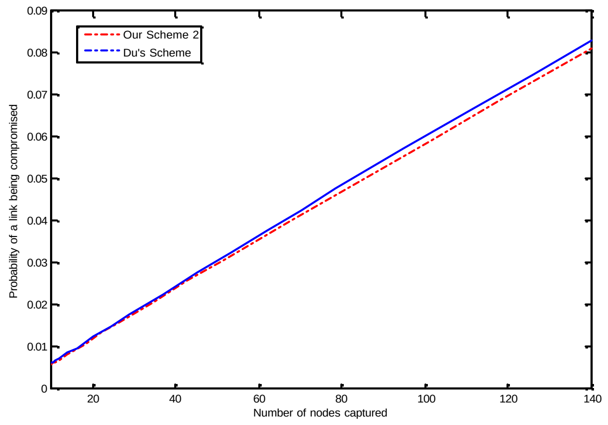

Fig. 4. Number of nodes captured vs. probability of a link compromised. m=60, s_i=2, s_c=2

4.3 Path Key Establishment Overhead

As the number of hops in path key establishment phase increases, a node can reach more of its neighbors and communication cost increases. We analyzed path key establishment through simulations for scheme II and depicted the results in Figure 6. The ratio of neighbors that a node can reach in *i* hops is defined as *pl(i)*. Obviously, *pl(1)* gives local connectivity. Our scheme performs better than Du et al.'s scheme [3] such that our scheme needs less number of hops for small *m* values. It can be observed from Figure 6 that for *m*=60 or larger values of *m*, a node can reach all its neighbors in at most two hops. In [3], only 63% of the nodes reach their neighbors in at most two hops when *m*=60. Moreover, in [3], *m* should be 200 in order for a node to reach all of its neighbors in at most two hops.

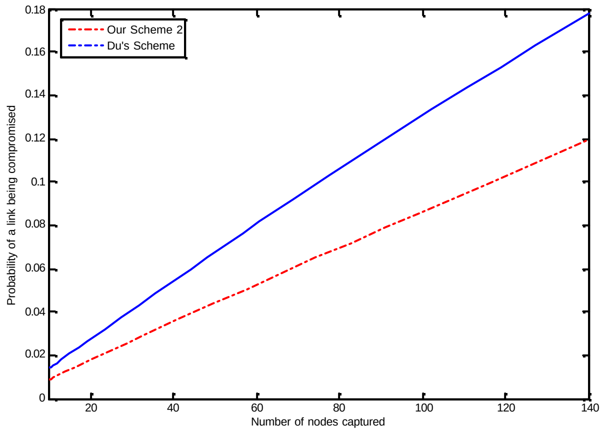

Fig. 5. Number of nodes captured vs. probability of a link compromised for p_{local}=0.86. For our scheme: m=90, s_i=3 and s_c=3. For Du's scheme: m=140.

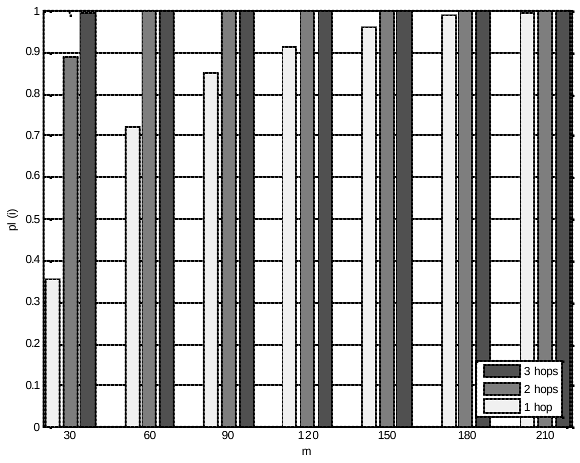

Fig. 6. Path Key Establishment Overhead

5 Conclusions

In this paper, we proposed a new deployment model and two novel key predistribution schemes based on the proposed model. In our deployment model, we proposed the nodes to be deployed in lines in a continuous fashion. This model is practical and can be realized easily. In the proposed scheme I, we assume to know the deployment points of each node and with that knowledge we distribute pairwise keys to each node to be used for communication between its neighbors. In scheme II, we loosen this assumption and assume that a node can be at any deployment point in a known line.

We compared our schemes with Du et al.'s key predistribution scheme [3]. Performance evaluation showed that scheme I can reach high local connectivity values even with small memory usage. This is due to the assumption in scheme I that we can know the neighbors of each node according to their deployment points. However, there is an upper limit in local connectivity; other schemes can have better local connectivity with high memory usage, whereas local connectivity in scheme I stays the same at 0.85 after a certain point. On the other hand, scheme II achieves higher local connectivity values than Du's scheme in all cases. Both scheme I and II show good performance in global connectivity and it is possible to reach 100% global connectivity with small memory usage. Moreover, scheme II has better node capture resiliency than Du et al.'s scheme with the same local connectivity value. Furthermore, communication cost of path key establishment overhead is smaller in our schemes.

References

[1] 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.

[2] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Research in Security and Privacy*, pages 197–213, 2003.

[3] W. Du, J. Deng, Y. S. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *Proceedings of IEEE INFOCOM'04*, March 2004.

[4] W. Du, J. Deng, Y. S. Han, and P. Varshney. A pairwise key predistribution scheme for wireless sensor networks. In *Proceedings of ACM CCS'03*, pages 42–51, October 2003.

[5] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the ACM CCS'02*, pages 41–47, November 2002.

[6] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of ACM CCS'03*, pages 52–61, October 2003.

[7] D. Liu and P. Ning. Location-based pairwise key establishments for static sensor networks. In *Proceedings of ACM SASN '03*, pages 72–82, October 2003.

[8] S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of ACM CCS'03*, pages 62–72, October 2003.

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

[10] D. Liu, P. Ning, and W. Du. Group-Based Key Pre-Distribution in Wireless Sensor Networks. In *Proceedings of 2005 ACM Workshop on Wireless Security*.

[11] D. Huang, M. Mehta, D. Medhi, and L. Harn. Location aware Key Management Scheme for Wireless Sensor Networks. *SASN'04*, October 25, 2004, Washington, DC, USA.

[12] R. Blom. An optimal class of symmetric key generation systems. In *Proceedings of EUROCRYPT 84*, 1985.