

ECPKS: An Improved Location-Aware Key Management Scheme in Static Sensor Networks

Ashok Kumar Das

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur 721 302, India (Email: akdas@cse.iitkgp.ernet.in)

(Received June 27, 2006; revised and accepted Dec. 05, 2006)

Abstract

A location-aware scheme uses a priori knowledge of the deployed sensor nodes of some target field in a sensor network. Such location-aware schemes improve substantially higher network connectivity and resilience against node captures if the deployment error between the actual and expected locations of the deployed sensor nodes is smaller. The ideas of using pre-deployment and post-deployment knowledges for the key pre-distribution techniques in a sensor network are two mutually orthogonal techniques. Therefore, it may be desirable to combine them together to provide better performances. In this paper, we propose a scheme which takes advantages of both pre-deployment and post-deployment knowledges of the deployed sensor nodes in a sensor network. Our scheme significantly improves the performance of establishing pairwise keys between neighbor sensor nodes than the existing previous key pre-distribution schemes. Moreover, it provides a better trade-off among network connectivity, communication overhead, computational overhead, storage requirement and security against node capture than the existing schemes. In addition, it supports addition of new sensor nodes after the initial deployment of sensor nodes and also works for any deployment topology.

Keywords: Key pre-distribution, location-aware key pre-distribution schemes, post-deployment knowledge, pre-deployment knowledge, security, sensor networks

1 Introduction

A sensor network consists of many tiny computing nodes called sensors, that are scattered in a target field in order to sensing some information and transmitting the information to nearby base stations for further processing. A sensor network is different from a traditional distributed network in various aspects. To make sensor networks economically viable, sensor devices are limited in their energy, computation, and communication capabilities. Each sensor node contains a primitive processor featuring very low computing speed and only small amount of programmable memory. A sensor node is battery pow-

ered and is expected to operate only for few days. Sensor nodes have the ability to communicate with each other and with the base station by short range wireless radio transmission at low bandwidth and over small communication ranges (typically 30 meters). A survey on sensor networks could be found in [1, 2].

Sensor networks are widely deployed in a variety of applications ranging from military to environmental and medical research. Mostly, for military applications, information collected by sensor nodes need be encrypted before transmission. Due to resource limitations of sensor nodes, it is not practical to use public key cryptographic technique such as RSA [16] or Diffie-Hellman Key Exchange Protocol [5] or Elliptic Curve Cryptography (ECC) [18]. A symmetric cipher such as DES [18] or RC5 [17] or IDEA [18] or AES [4] is the viable option for encryption or decryption of secret information. However, establishing symmetric keys among communicating nodes is a challenging problem. Key materials are pre-distributed in each sensor node's memory by a (key) set-up server before the deployment of the nodes. After the deployment, each node finds out the neighbors with which it can communicate using the pre-distributed keys. This key establishment protocol is referred as *bootstrapping protocol*.

In wireless sensor networks (WSNs), the following general securities are required:

- *Authentication:* authenticating the other sensor nodes and the base stations before granting a limited resource, or revealing information.
- *Integrity:* ensuring that message or the entity under consideration is not altered.
- *Confidentiality:* providing privacy of the wireless communication channels to prevent eavesdropping.
- *Non-repudiation:* preventing the malicious nodes to hide their activities.

These above security requirements can be provided by a key pre-distribution mechanism with the following requirements:

- *Scalability:* ability to support large networks. This means that the key pre-distribution algorithms must

support large networks, and must be flexible against substantial increase in the size of the network even after the deployment of the sensor nodes in a sensor network.

- *Efficiency*: storage, processing and communication limitations on sensor nodes must be less.
- *Network connectivity*: probability that two sensor nodes share the same key or keying material. Enough network connectivity must be provided for a WSN in order to perform its intended functionality.
- *Resilience against node capture*: resilience measurement against node capture is computed by comparing the number of nodes captured, with the fraction of total network communications that are exposed to the adversary *not including* the communications in which the compromised nodes are directly involved.

A *location-adaptive* key pre-distribution scheme may or may not use any prior knowledge of the deployment locations of the sensor nodes, but it can adapt to the geography of deployment areas. On the other hand, a *location-aware* key pre-distribution scheme depends on the prior knowledge of the deployment locations like the expected location of each sensor and the overall geometry of the deployment area. This pre-deployment knowledge helps us to tune the key pre-distribution algorithms to achieve better network connectivity and higher resilience against node capture.

Several techniques [3, 6, 7, 8, 12, 13, 14] are proposed in the literature to solve the bootstrapping problem. Eschenauer and Gligor proposed the first basic random key predistribution scheme popularly known as the EG scheme [8]. In the key predistribution phase, the (key) setup server selects a large pool of randomly generated key materials where each key material is simply a randomly generated number. For each sensor node to be deployed in the network, a subset of m keys from the key pool is selected *randomly without replacement* and loaded in sensor's memory. This subset is called the key ring of a sensor node. After deployment of sensor nodes, the direct key establishment phase (also called shared key discovery phase) is performed by the deployed sensor nodes. In this phase, each node first discovers its physical neighbors which are within its communication range. Then, if two neighbor sensor nodes, say, u and v share a common key, they become key neighbors. The path key establishment phase is an optional phase, and if executed, adds to the connectivity of the network. If two neighbor nodes, say, u and v fail to establish a direct key between them in the direct key establishment phase, they find a secure h -hop path between them, and one of them, say, u generates a new secret pairwise key between them and finally transmits that key securely along the discovered path to the desired destination node v . The communication overhead increases as the number h of hops of the secure path increases. However, in practical situations, the number h

of hops is restricted to a small value, say, 2 or 3. From the analysis of the EG scheme, it follows that connectivity of the network depends on size of the key pool. If the key pool size is chosen smaller, it leads to higher network connectivity. Since the key rings are selected *randomly without replacement*, the same key may be repeated for several pair of neighbor nodes in the network. As a result, even if the number of captured nodes is small, they can reveal a large fraction of total communications in the network which in turn degrades the security.

Chan et al. [3] proposed several extended versions of the EG scheme which are the q -composite scheme, the multi-path key reinforcement scheme and the random pairwise keys scheme. All these schemes improve the security significantly compared to the EG scheme. However, the multi-path key reinforcement scheme requires more communication overhead to establish a pairwise key between two neighbor sensor nodes. The polynomial pool-based key pre-distribution scheme [12] proposed by Liu and Ning and the matrix-based key distribution scheme [6] proposed by Du et al. improve the performances significantly.

The rest of this paper is organized as follows. We give a brief overview of existing key pre-distribution schemes in Section 2. In Section 3, we introduce our proposed scheme which is based on deployment knowledges of the sensor nodes. In Section 4, we provide a detailed theoretical analysis of our scheme with respect to network connectivity, resilience against node capture, communication overhead, computational overhead and storage requirement. In Section 5, we compare the performances of our scheme with the existing schemes. Finally, we conclude the paper in Section 6.

2 Background: Overview of Existing Key Pre-Distribution Schemes

In this section, we discuss the random pairwise keys scheme [3] proposed by Chan et al and the polynomial-pool based scheme [12] proposed by Liu and Ning. We also give an overview of two existing location-aware schemes, namely, the closest pairwise keys scheme (CPKS) [14] and the closest polynomials pre-distribution scheme (CPPS) [14].

2.1 The Random Pairwise Keys Scheme

Let m be the size of the key ring of each sensor node and p the probability that any two nodes be able to communicate securely. In the key predistribution phase, a total of $n = \frac{m}{p}$ unique node identifiers are generated. The actual size of the network may be smaller than n . For each sensor node to be deployed, a set of m other randomly selected distinct node ids and a pairwise key is generated for each pair of nodes. The key is stored in both nodes'

key rings along with the id of the other node that also knows the key.

In the direct key establishment phase, each node broadcasts their own ids to their neighbor nodes in communication ranges. Two neighbor nodes can then easily verify the id of a neighbor node in their key rings. If the id of a neighbor node is found in a node's key ring, they share a common pairwise key for communication. A cryptographic handshake is then performed between neighbor nodes for mutual verification of the common key.

Since the pairwise key between two nodes is generated randomly, no matter how many nodes are captured by an adversary, the other non-compromised nodes communicate with each other with 100% secrecy. Thus, the random pairwise keys scheme provides unconditional security against node capture attacks.

2.2 The Closest Pairwise Keys Scheme (CPKS)

2.2.1 Description of CPKS

The closest pairwise keys scheme (CPKS) [14] proposed by Liu and Ning is a location-aware scheme. It is the extended version of the random pairwise keys scheme which uses priori deployment knowledge of the deployed sensor nodes.

In the key pre-distribution phase, for each sensor node u to be deployed in the target field, the key setup server determines a set S of c other nodes whose expected locations of deployment are closest to that of u . Let $S = \{v_1, v_2, \dots, v_c\}$. For every $v_i \in S$ ($i = 1, 2, \dots, c$) for which a pairwise key between u and v_i has not already been assigned by the setup server, a new random symmetric key k_{u,v_i} is generated. The key-plus-id combination (k_{u,v_i}, v_i) is loaded to u 's key ring, whereas the pair (k_{u,v_i}, u) is loaded to v_i 's key ring. Thus, we note that before deployment of the sensor nodes in the deployment field, each sensor node u is loaded with c key-plus-id combinations $\{(k_{u,v_i}, v_i), i = 1, 2, \dots, c\}$ into its key ring.

After deployment of the sensor nodes in a deployment field, they start the direct key establishment phase (the shared key discover phase) in order to establish secret keys between neighbor sensor nodes. At first, each sensor node locates its physical neighbors in its communication range. Two neighbor nodes, say, u and v can establish a secure communication link, if they share a pre-distributed pairwise key in their key rings. Nodes u and v exchange their identifiers to each other. If the id of u is resident in v 's key ring as well as the id of v is also resident in u 's key ring, both nodes u and v use their common key $k_{u,v}$ in their key rings for future communication with each other. Thus, we note that to identify a common key is trivial in this phase, because each pairwise key in a particular node is accompanied by the id of the other nodes holding the key. A cryptographic handshake may be performed between neighbor nodes u and v for mutual verification of the common key possessed by them.

2.2.2 Network Connectivity of CPKS

Assume that the deployment field is two dimensional. Let u be a sensor node whose expected location be (u_x, u_y) , whereas its actual location be (u'_x, u'_y) . This corresponds to a deployment error $e = (u'_x - u_x, u'_y - u_y)$. Liu and Ning showed that the network connectivity of CPKS depends upon the deployment error e . If the maximum deployment error e is small, CPKS provides significantly better connectivity than the random schemes [3, 8, 12]. They have also shown that for sufficiently large errors, CPKS essentially degrades to the random pairwise keys scheme [3] which has very poor connectivity when the network size is larger.

2.2.3 Security of CPKS

We note that each pre-distributed pairwise key $k_{u,v}$ between two neighbor nodes u and v is randomly generated. Thus, no matter how many nodes are captured, the pairwise keys between non-compromised sensor nodes remain still secure. This means that no matter how many sensor nodes are captured, the non-compromised nodes can communicate with each other with 100% secrecy. In this way, CPKS provides unconditional security against node capture attacks.

2.3 The Polynomial-Pool Based Key Pre-Distribution Scheme

Liu and Ning's polynomial-pool based key distribution scheme [12] can be described as follows. Let $F_q = GF(q)$ be a finite field with a q (either a prime or 2^m for some positive integer m) just big enough to accommodate a symmetric cryptographic key. Let $f(x, y) \in F_q[x, y]$ be a t -degree bivariate symmetric polynomial i.e., $f(x, y) = f(y, x)$. The coefficients of the polynomial $f(x, y)$ are chosen from the finite field F_q . A *polynomial share* of $f(x, y)$ is a univariate polynomial $f(u, y)$ for some $u \in F_q$. We have, $f(u, v) = f(v, u)$.

Thus, if two shares $f(u, y)$ and $f(v, y)$ of the same polynomial $f(x, y)$ are given to two nodes, say, u and v , they can come up with the common value $f(u, v) \in F_q$ as a shared key between them. If $(t + 1)$ or more shares of $f(x, y)$ are known, one can easily reconstruct $f(x, y)$ uniquely using the *Lagrange's Interpolation* [9]. Thus, the disclosure of up to t shares does not reveal the polynomial $f(x, y)$ to an adversary and uncompromised shared keys based on $f(x, y)$ remains completely secure.

The key setup server selects a random pool \mathcal{K} of s symmetric bivariate polynomials in $F_q[x, y]$ each of degree t in x and y . Some ids $u_1, u_2, \dots, u_n \in F_q$ are also generated for the sensor nodes in the network, where n is the network size. For each sensor node u to be deployed in the network, s' polynomials, say, $f_1(x, y), f_2(x, y), \dots, f_{s'}(x, y)$ are randomly selected from \mathcal{K} and the polynomial shares $f_1(u, y), f_2(u, y), \dots, f_{s'}(u, y)$ are loaded in the key ring K_u of u . Immediately after deployment, each sensor u transmits the ids of the polynomial shares residing in its

key ring. Two physical neighbors u and v having shares of some common polynomial(s) can establish a pairwise key between them.

2.4 The Closest Polynomials Pre-Distribution Scheme (CPPS)

2.4.1 Description of CPPS

The closest polynomials pre-distribution scheme (CPPS) [14] proposed by Liu and Ning is a location-aware scheme. It is based on the polynomial-pool based scheme which uses priori deployment knowledge of the deployed sensor nodes. The deployment field is partitioned into rectangular cells $C_{i,j}$ containing R rows and C columns. The four adjacent neighboring cells of a cell $C_{i,j}$ are considered as the cells $C_{i,j-1}$, $C_{i-1,j}$, $C_{i,j+1}$, and $C_{i+1,j}$.

In the key predistribution phase, the key setup server chooses $s = R \times C$ random symmetric t -degree bivariate polynomials $f_{i,j}(x, y) \in F_q[x, y]$, $i = 1, 2, \dots, R$, $j = 1, 2, \dots, C$, where $F_q = GF(q)$ is a finite field with q large enough to accommodate a cryptographic key. Here $GF(q)$ stands for the Galois field of order q where q is either a prime or of the form 2^m for some positive integer m . Assume that the expected deployment location of a node u lies in the cell $C_{i,j}$ called the *home cell* of u . The key ring of u is loaded with the shares of the five polynomials corresponding to the home cell and the four neighboring cells. That is, u is given the five polynomial shares: $f_{i,j}(u, y)$, $f_{i-1,j}(u, y)$, $f_{i+1,j}(u, y)$, $f_{i,j-1}(u, y)$, $f_{i,j+1}(u, y)$. The key set-up server also stores in u 's memory the id (i, j) of its home cell. Thus, we note that each sensor node's key ring contains five t -degree polynomial shares before its deployment in the target field. Since each t -degree polynomial share consists of $(t+1)$ coefficients which are from F_q , so the storage requirement for this polynomial share is $(t+1) \log q$ bits. Hence, the storage requirement for each sensor node is $5(t+1) \log q$ bits in order to store its keying informations. If each symmetric key is taken of q bits, the storage requirement is equivalent to store $5(t+1)$ symmetric keys.

In direct key establishment phase, each node u broadcasts its home cell's id (i, j) (or some messages encrypted by potential pairwise keys) to its immediate physical neighbors. Suppose u and v be two neighbors. From the co-ordinate of the home cell (i, j) of the source node u , the destination node v can immediately determine the ids of the polynomial shares the source node u has. Let u and v find a common polynomial, say, f . Then the source node u computes a common pairwise key shared with the destination node v as $f(u, v)$. Similarly, the destination node v also computes a common pairwise key shared with the source node u as $f(v, u)$. Since $f(u, v) = f(v, u)$, they store this value $f(u, v)$ as the secret key shared between them for their future communication. A cryptographic handshake may be performed between neighbor nodes u and v for mutual verification of the common key $f(u, v)$ possessed by them.

2.4.2 Network Connectivity of CPPS

Similar to the analysis of CPKS, the network connectivity of CPPS also depends on the deployment error. Larger error leads to poorer connectivity. Moreover, the network connectivity of CPPS depends on the length L of cell side. Liu and Ning showed that if L is chosen larger, there is a higher probability of establishing a direct key between two neighbors. Thus, smaller L leads also to poorer connectivity. If the maximum deployment error is small, CPPS also provides significantly better connectivity than the random schemes [3, 8, 12].

2.4.3 Security of CPPS

As long as no more than t polynomial shares of a bivariate polynomial are disclosed, an attacker knows nothing about the non-compromised pairwise keys established using this polynomial. Thus, the security of this scheme depends on the average number of nodes sharing the same polynomial, or equivalently on the number of nodes that are expected to be located in each cell and its four adjacent cells. If that number is larger than t , CPPS is not unconditionally secure. Again, in CPPS, if the length L of cell side is larger, it leads to a larger number of sensor nodes sharing the same bivariate polynomial, which in turn degrades the security performance. As a result, CPPS is unconditionally secure and t -collusion resistant.

3 The Proposed Scheme

In this section, we describe the main motivation behind our proposed scheme. We also describe various phases to this scheme.

3.1 Motivation

Our scheme is motivated by the following considerations. The location-aware schemes such as CPKS and CPPS described above, lose their performance enhancements as the deployment error between the actual and expected locations of the deployed sensor nodes increases. For sufficiently larger errors such location-aware schemes essentially degrade to a random pairwise keys scheme [3] without a priori knowledge of deployment configuration. The ideas of using both pre-deployment as well as post-deployment knowledges for the key pre-distribution mechanisms in a sensor network are mutually orthogonal techniques. It may be desirable to combine them together to provide better performances. In this paper, we introduce a scheme which is based on both pre-deployment and post-deployment knowledges of the deployed sensor nodes. In order to achieve this goal, we use the closest pairwise keys scheme (CPKS) to take advantages of pre-deployment knowledge of deployed sensor nodes. However, in practical situations, it is not always possible to deploy sensor nodes in a target field to their expected

locations. As a result, there will be always a deployment error between the actual and the expected locations of the deployed sensor nodes. From the analysis of CPKS, we observe that the network connectivity degrades if the maximum deployment error increases. In order to further improve performances we take the advantages of post-deployment locations of deployed sensor nodes. Due to [14], it is a possible task for sensor nodes to determine their post-deployment locations securely after their initial deployment. To take advantages of post-deployment knowledges of deployed sensor nodes, we use the key prioritization technique [14] applied over the basic probabilistic scheme (the EG scheme) [8]. Hence, our proposed scheme is an enhancement of CPKS in conjunction with the key prioritization technique applied over the EG scheme. We call our scheme as the *enhanced closest pairwise keys scheme* (ECPKS). ECPKS works for any deployment topology and there is no need to assume the deployment area as rectangular (as in CPPS [14]).

The different phases of ECPKS are as follows.

3.2 Key Pre-Distribution

This phase is performed by the key setup server in offline before deployment of the sensor nodes in some target field. It consists of the following steps:

Step 1. Let n be the size of the sensor network. For each sensor node u , the key setup server assigns a unique identifier id_u .

Step 2. The key setup server selects a large key pool \mathcal{K} of M key units. Each key unit consists a random number (i.e., a symmetric cryptographic key) generated by the setup server and a unique location (x, y) in the deployment field associated with it. For example, a key unit for a sensor node i is of the form $ku_i = (k, (x_i, y_i))$ where k is a randomly generated symmetric key by the (key) setup server and (x_i, y_i) is the location attached with k . For convenience we refer this location (x_i, y_i) as the id of the key k .

Step 3. For each sensor node u to be deployed in the sensor network, the setup server selects a set S_1 of c sensor nodes whose expected locations are closest to sensor node u . For each sensor node v in S_1 , for which a pairwise key between u and v has not already been generated, a new random key k_{uv} is generated. The key-plus-id combination (k_{uv}, id_v) is loaded to u 's key ring, whereas the pair (k_{uv}, id_u) is loaded to v 's key ring.

Step 4. To further improve performances of our scheme, we load an excessive amount of key materials in each sensor node initially. Crossbow Technology Inc. [10] develops a typical MICA2 mote sensor device which has 512KB EEPROM, but only 4KB RAM. Thus, it is practical to store more pre-distributed keying information in EEPROM of a sensor device. Since

the low priority key units after the key prioritization phase will be deleted from memory, so the returned memory will be used for application part by the sensor node. To achieve this goal, for each sensor node u to be deployed in the network, the setup server selects another set S_2 of m key units randomly from the key pool \mathcal{K} and loads this set to u 's key ring.

As a result, we notice that the key ring KR_u of each sensor node u contains $(c + m)$ key units in its memory before its deployment in the target field.

3.3 Direct Key Establishment

Immediately after the deployment, each sensor node first locates its all neighbors in its communication range. Two nodes are called *physical neighbors* if they lie in each other's communication range. They are called *key neighbors* if they share a common key. They are called *direct neighbors* if they are both physical and key neighbors. After the deployment of the sensor nodes, two physical neighbor nodes, say, u and v can establish a secure communication link, if they share a pre-distributed pairwise key. To achieve this, we use the first c key units in each node's key ring. To identify a common key between u and v becomes trivial in this case, because each node can easily verify the id of other nodes in its c key units. Thus, sensor nodes deployed to their expected locations in the target field are able to establish pairwise keys between them. *The remaining unused key units from these c key units will be deleted from their key rings so that the returned memory will be used for application part.*

Communication Steps:

This phase is summarized below. $MAC_k(M)$ refers to the message authentication code (MAC) for the message M , under the key k . We refer RN_u as a random nonce generated by a sensor node u . $A||B$ refers that data A concatenates with data B .

In order to establish a secret key between two neighbor nodes, say, u and v , they need to exchange the following messages:

- 1) Node u generates a random nonce RN_u . It then sends its own id id_u and RN_u to its neighbor v .
 $u \rightarrow v : id_u || RN_u$.
- 2) Node v also generates a random nonce RN_v and then sends its own id id_v as well as RN_v to its neighbor u .
 $v \rightarrow u : id_v || RN_v$.
- 3) Assume that the id of v is found from first c key units of u 's key ring. Let k_{uv} be the key shared with node v corresponding to that id. It sends its own id id_u , the id id_v of v as well as the random nonce RN_v of v and a message authentication code (MAC) on these fields under the key k_{uv} to node v .
 $u \rightarrow v : T = (id_u || id_v || RN_v) || MAC_{k_{uv}}(T)$.

- 4) In a similar fashion, v verifies the id of u in first c key units of its key ring and assume that v finds a secret key k_{uv} shared with node u corresponding to that id. It then sends its own id id_v , the id id_u of u as well as the random nonce RN_u of u and a message authentication code (MAC) on these fields under the key k_{uv} to node u .
- $$v \rightarrow u : T' = (id_u || id_v || RN_u) || MAC_{k_{uv}}(T').$$

After receiving the last message by the nodes u and v , they perform one symmetric cryptographic MAC verification on that message, under the key k_{uv} . If the MAC verification is successful, then only they store the key k_{uv} for their future communications. We note that by sending the random nonces of each other by the nodes u and v , the transaction between them is determined uniquely.

3.4 Key Prioritization

If the maximum deployment error increases, the network connectivity gradually degrades. Assume that two physical neighbors do not able to establish a pairwise key in direct key establishment phase. In order to establish a pairwise key between two neighbor nodes u and v , they use their post-deployment locations. We assume that every sensor node in the sensor network can discover its real post-deployment location securely after the deployment of the sensor nodes. Akyildiz et al. [1] pointed out that “most of the sensing tasks require knowledge of positions,” and “location finding systems are required by many of the proposed sensor network routing protocols”. There are also recent development in determining individual sensor nodes’ positions either with a global positioning system (GPS) [11] or local references [15]. Thus, it is practical to assume for the sensor nodes to determine their post-deployment locations securely.

The main idea behind key prioritization is as follows. By using memory for sensing applications, a sensor node can keep a large number of key units during key predistribution phase. After deployment, once a sensor node determines its post-deployment location, it can prioritize these key units based on this post-deployment knowledge. A sensor node can then give up the key units that are less likely to be used for pairwise key establishment in order to thwart against node capture attacks by an adversary and return the memory to the sensing applications. Thus, it has a higher probability to keep those key units that may be required for secure communications with its neighbor nodes in the sensor networks. Hence, this phase reuses the memory for sensing applications to keep more key units during key predistribution phase, keeps the higher priority key units that are most likely to be used after the post-deployment location is known, and finally discards the low priority key units in order to thwart against node capture attacks and returns the memory to the applications.

This phase uses the last remaining m key units from the key rings of sensor nodes. This phase consists of the following steps:

Step 1. Sensor node u securely determines its post-deployment location, say $PDL_u = (u_x, u_y)$. u computes the distances between PDL_u and the locations attached to these m key units. Then u ranks these m key units in its key ring KR_u in increasing order according to the computed distances. The shorter distance has the higher priority. Node u chooses c' highest priority key units from these key units and deletes the remaining $(m - c')$ key units from its key ring which are less likely to be used for pairwise key establishment in order to thwart against node capture attacks. The returned memory is used for application by the sensor node u .

Step 2. Similarly, a physical neighbor v of sensor node u performs the above step.

Step 3. It is easy to observe that if two nodes are close to each other, there is a high probability to establish a direct key between them. To establish a direct key between them, nodes u and v exchange only the locations (ids) of the c' highest priority key units from their key rings.

Communication Steps:

In order to establish a secret key between two neighbor nodes u and v , the following messages are to be exchanged between them:

- 1) Node u sends its own id, a randomly generated nonce RN_u and a list of c' highest priority key ids to the node v :

$$u \rightarrow v : id_u || RN_u || \{ \text{list of } c' \text{ highest priority key ids} \}.$$
- 2) Similarly, node v also sends its own id, a randomly generated nonce RN_v and a list of c' highest priority key ids to the node u :

$$v \rightarrow u : id_v || RN_v || \{ \text{list of } c' \text{ highest priority key ids} \}.$$
- 3) Assume that u finds q common keys, say, k_1, k_2, \dots, k_q ($q \geq 1$) from the c' highest priority key units. It computes a secret key shared with node v as $k_{uv} = k_1 \oplus k_2 \oplus \dots \oplus k_q$ and then sends the following message with a MAC under the computed key k_{uv} to node v :

$$u \rightarrow v : T_1 = (id_u || id_v || RN_v) || MAC_{k_{uv}}(T_1).$$
- 4) Similarly, node v also computes the secret key k_{uv} shared with the node u . It then sends its own identifier id_v , node u 's identifier id_u as well as the random nonce RN_u of node u along with a message authentication code (MAC) of these fields under the key k_{uv} to node u :

$$v \rightarrow u : T_2 = (id_u || id_v || RN_u) || MAC_{k_{uv}}(T_2).$$

Finally, both nodes u and v have to perform one MAC verification on the last message received by them under the key k_{uv} . If the MAC verification is successful, they store this key k_{uv} for their future communications. We

also note from this phase that the transaction is determined uniquely because each node attaches other node's nonce to its message.

3.5 Dynamic Node Addition

During the lifetime of a sensor network, some nodes may be damaged or compromised by an adversary. Therefore, new sensor nodes may be added to replace such nodes in the network. For doing this, the setup server performs the above key pre-distribution phase for the new sensor node to be deployed and then informs the deployed sensor nodes chosen for the new sensor node the corresponding keys through the secure channels. To establish direct keys with its physical neighbors, it performs direct key establishment and key prioritization phases as described above.

4 Analysis of the ECPKS Scheme

In this section, we derive the probability of establishing direct keys between neighbor sensor nodes and analyze the security issues for our proposed scheme (ECPKS). We also analyze storage requirement, communication overhead and computational overhead required for our scheme.

4.1 Probability of Establishing Direct Keys between Neighbors

We use the following notations for deriving the probability of establishing a direct key between two neighbor nodes. We also use those notations for discussion in rest of this paper.

- n : the network size.
- d : the average number of neighbor sensor nodes of each sensor node.
- c : the number of pre-distributed key units to each node for the direct key establishment phase of ECPKS.
- M : the key pool size.
- m : the number of pre-distributed key units to each node for the key prioritization phase of ECPKS.
- ρ : the communication range.
- e : the maximum deployment error.
- E_1 : the event that a sensor node can establish a key during the direct key establishment phase of ECPKS with one of its d physical neighbors using only the first c key units from its key ring.
- p_1 : $P(E_1)$, the probability that two neighbors establish a direct key for the event E_1 .

- E_2 : the event that a sensor node can establish a direct key during the key prioritization phase of ECPKS with one of its d' physical neighbors, where $d' = \lfloor (1 - p_1) \cdot d \rfloor$, using the last m key units from its key ring.
- p_2 : $P(E_2)$, the probability that two neighbors establish a direct key for the event E_2 .
- p : the overall probability that two neighbors establish a direct key which is same as the probability that at least one of the events E_1 and E_2 will occur.

It is easy to observe that the events E_1 and E_2 are stochastically independent. Hence, we have $P(E_1 \cap E_2) = P(E_1) \cdot P(E_2)$. Now, $P(\bar{E}_1 \cap \bar{E}_2) = 1 - P(E_1 \cup E_2) = 1 - [P(E_1) + P(E_2) - P(E_1 \cap E_2)] = (1 - P(E_1))(1 - P(E_2)) = P(\bar{E}_1)P(\bar{E}_2)$, where $P(\bar{E})$ represents the probability of the complement of the event E . Thus, both the events \bar{E}_1 and \bar{E}_2 are also stochastically independent. As a result, the required probability that at least one of the events E_1 and E_2 will occur is $1 - (\text{probability that none of the events } E_1 \text{ and } E_2 \text{ will occur}) = 1 - P(\bar{E}_1 \cap \bar{E}_2) = 1 - (1 - p_1)(1 - p_2) = p_1 + p_2 - p_1 p_2$. Therefore, we obtain the overall probability of establishing a direct pairwise key between two neighbor sensor nodes as

$$p = p_1 + p_2 - p_1 p_2.$$

For the derivation of p_1 , we use only the first c key units from the key ring of each sensor node. For the sake of simplicity, we assume that the target field is two-dimensional, so that every point in that region is expressed by two coordinates x and y . Assume that u is a sensor node whose expected location is (u_x, u_y) whereas its actual location is (u'_x, u'_y) . This corresponds to a deployment error of $e_u = (u'_x - u_x, u'_y - u_y)$. The actual location (or equivalently the error e_u) can be modeled as a continuous random variable that can assume values in R^2 . The probability density function $f_u(u'_x, u'_y)$ of (u'_x, u'_y) characterizes the pattern of deployment error. As in [13] we assume that (u'_x, u'_y) is uniformly distributed within a circle with center at (u_x, u_y) and radius e called the *maximum deployment error*. We have:

$$f_u(u'_x, u'_y) = \begin{cases} \frac{1}{\pi e^2} & \text{if } (u'_x - u_x)^2 + (u'_y - u_y)^2 \leq e^2; \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

As in [7] another way to model (u'_x, u'_y) as a random variable following the two-dimensional normal (Gaussian) distribution with mean (u_x, u_y) and variance σ^2 . The corresponding probability density function is:

$$f_u(u'_x, u'_y) = \frac{1}{2\pi\sigma^2} e^{-[(u'_x - u_x)^2 + (u'_y - u_y)^2]/(2\sigma^2)}.$$

However, for the sake of simplicity we only consider the uniform distribution given in Equation (1).

Let u and v be two deployed nodes. Assume that the different nodes are deployed independently i.e., (u'_x, u'_y) and (v'_x, v'_y) are independent random variables. Since u

and v are independently deployed, the probability that u and v are in each other's communication range is given by

$$p_1(u, v) = \iiint_{C_1} f_u(u'_x, u'_y) f_v(v'_x, v'_y) du'_x du'_y dv'_x dv'_y,$$

where C_1 is the region $(u'_x - v'_x)^2 + (u'_y - v'_y)^2 \leq \rho^2$.

Let us consider that the key neighbors of u are uniformly distributed in a circle of radius ρ' . Since u can share pairwise keys with c nodes, the expected value of ρ' is given by $\rho' = \rho \times \sqrt{\frac{c}{d+1}}$. Let v be a key neighbor of u . Then, the probability that v lies in the physical neighborhood of u is given by

$$p_1(u) = \frac{1}{\pi\rho'^2} \iint_{C_2} p_1(u, v) dx dy,$$

where C_2 is the region $(x - u_x)^2 + (y - u_y)^2 \leq \rho'^2$. Again, since u is expected to have $c \times p_1(u)$ direct neighbors, the probability that u can establish a pairwise key with one of its physical neighbor is given by

$$p_1 = \frac{p_1(u) \times c}{d} \approx p_1(u) \times \lambda,$$

where $\lambda = \frac{c}{d+1}$. We take the communication range ρ as the basic unit of distance measurement, i.e., $\rho = 1$. One can compute the probability p_1 for the density function given in Equation (1) and establish that $p_1 \approx 1$ for small deployment errors.

For the derivation of p_2 , we use only the last remaining m key units from the key rings of sensor nodes during the key prioritization phase. Since a sensor node u can establish a direct key with its physical neighbor v with probability p_1 , so it could not able to establish pairwise keys till now with d' physical neighbors, where $d' = \lfloor (1 - p_1) \cdot d \rfloor$.

Let A be the area of the deployment field. Since a node u gets m keys during the key pre-distribution phase and ranks only c' highest priority keys after the key prioritization phase, so these c' keys' associated locations are no more than a distance r' away from u , where $r' = \sqrt{\frac{c' \times n}{m \times (d+1)}}$. The overlapping area between two neighbor nodes u and v is shown in Figure 1. B and C are the expected locations of the nodes u and v respectively. Assume that the distance between u and v is x . We observe that $BDCE$ is a rhombus. Let $\angle DBC = \theta$. We then have $\angle EBC = \angle DCB = \angle ECB = \theta$. Here $BD = DC = CE = BE = r'$ and $BC = x$. We also have, $BF = FC = \frac{x}{2}$. From the triangle BDF , we obtain, $\cos \theta = \frac{x}{2r'}$. Again, we have $DF = EF = \sqrt{r'^2 - \frac{x^2}{4}}$. Let d_1 be the length of first diagonal BC of the rhombus $BDCE$ and d_2 the length of the second diagonal DE of that rhombus. The area of the rhombus $BDCE$ is given by $\frac{1}{2} d_1 d_2 = x \sqrt{r'^2 - \frac{x^2}{4}}$. Thus, we obtain the required overlapping area $A(x)$ between nodes u and v as sum of the areas of the sectors BDE and CDE minus the area of the rhombus

$BDCE$ which is given by $A(x) = 2 \times \frac{\pi r'^2}{2\pi} \times 2\theta - x \sqrt{r'^2 - \frac{x^2}{4}}$. Hence, the overlapping area within both nodes' (i.e., u and v) communication radii is given by

$$A(x) = 2r'^2 \cos^{-1} \left(\frac{x}{2r'} \right) - x \sqrt{r'^2 - \frac{x^2}{4}}.$$

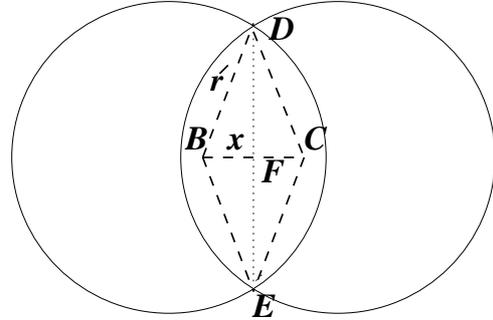


Figure 1: Overlapping area between two neighbor nodes u and v

The (average) number of pre-distributed keys that fall into $A(x)$ is given by

$$N'(x) = \lfloor m \times \frac{A(x)}{A} \rfloor,$$

and the total number of keys that are distributed over $A(x)$ is

$$N(x) = \lfloor M \times \frac{A(x)}{A} \rfloor.$$

Thus, the probability that two neighbor nodes u and v share a key during the key prioritization phase of ECPKS is computed as $1 -$ (probability that u and v do not share any keys) and hence, we have,

$$\begin{aligned} p_2(x) &= 1 - \frac{\binom{N(x) - N'(x)}{N'(x)}}{\binom{N(x)}{N'(x)}} \\ &= 1 - \prod_{i=0}^{N'(x)-1} \frac{N(x) - N'(x) - i}{N(x) - i}. \end{aligned}$$

As a result, the (average) probability of establishing a direct key between two neighbor sensor nodes during the key prioritization phase of ECPKS is given by

$$p_2 = \int_{r=0}^{\rho} \int_{\theta=0}^{2\pi} \frac{p_2(r)}{\pi\rho^2} r dr d\theta = 1 - 2 \int_0^1 g(r) dr,$$

where $g(r) = r \times \left(\prod_{i=0}^{N'(r)-1} \frac{N(r) - N'(r) - i}{N(r) - i} \right)$.

We have considered for the analysis of network connectivity the parameter values as follows:

- The communication range is $\rho = 30$ meters.
- The number of nodes in the network is $n = 10000$.

- The average number of neighbor nodes for each node is $d \leq 40$.
- The number of pre-distributed keys given for the direct key establishment phase is $c = 200$.
- The key pool size is $M = 40000$.
- The number of pre-distributed keys given for the key prioritization phase is $m = 100$.
- The number of highest priority keys after the key prioritization phase is $c' = 80$.
- The size of the deployment area A is chosen so that the maximum supported network size becomes $n = \lfloor \frac{A \times (d+1)}{\pi \times \rho^2} \rfloor$.

Figure 2 illustrates the relationship between the probability p of establishing a direct key between two neighbor nodes and the maximum deployment error e . From this figure it is very clear that $p \approx 1$ for small deployment errors. Moreover, this figure shows that ECPKS provides reasonable connectivity even for larger errors.

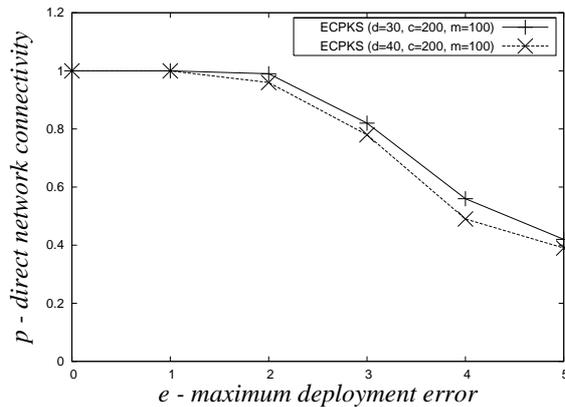


Figure 2: Direct network connectivity of ECPKS with $d = 30, 40$, $n = 10000$, $M = 40000$, $c = 200$, $m = 100$, and $c' = 80$

4.2 Security Against Node Capture

We observe that if the maximum deployment error is small, the network connectivity is high. Since ECPKS uses the first c key units for the direct key establishment phase, and these keys are different from each other because of the fact that they are generated randomly, so capturing of a node does not lead to compromise the secret communications between non-compromised nodes.

We note that if the sensor nodes are compromised before the key prioritization phase, they may reveal more secrets than compromising the same set of sensor nodes after the key prioritization phase. Thus, an adversary can observe this property and attack the network in the time period between the key pre-distribution and the key prioritization phases. This time period is known as the *window of vulnerability*. However, once a sensor node securely

determines its post-deployment location after its deployment in the target field, it can immediately complete the key prioritization. Thus, we believe the sensor nodes can be protected fairly well during the window of vulnerability. The most vulnerable period of time is the period after deployment and before key prioritization. However, this period is actually quite short due to the ease of completing key prioritization. As a result, we only consider the fact that no nodes are compromised during the window of vulnerability. We assume that an attacker (adversary) randomly compromises N_c sensor nodes in the network after the window of vulnerability.

Assume that there are n sensor nodes in the sensor network. This network is considered as an undirected graph with n vertices each having the same degree d , where d is the average number of neighbor nodes of each node. The total number of edges of the undirected graph with n vertices is $\frac{\sum_{i=1}^n d}{2} = \frac{nd}{2}$. Hence, the total direct communication links in the sensor network of size n is $\frac{nd}{2}$. Let us consider the direct key establishment phase of ECPKS. In this phase, each pairwise key between two neighbor nodes was generated by the key setup server randomly and thus, those keys are different for each pair of neighbor nodes in the network. Therefore, no matter how many nodes are compromised secret communications between non-compromised nodes are still secure. As a result, we have $\frac{nd}{2} \cdot p_1$ secure links so far even after capturing N_c nodes by the adversary. We now consider the key prioritization phase of ECPKS. Since a sensor node keeps only c' highest priority key units after key prioritization phase in order to establish secret keys with its remaining $(1 - p_1)d$ neighbor nodes, $\frac{nd}{2}(1 - p_1)p_2 \times [1 - (1 - \frac{c'}{M})^{N_c}]$ links will be compromised by the adversary from the total $[\frac{nd}{2}p_1 + \frac{nd}{2}(1 - p_1)p_2]$ secure links in the sensor network. Hence, the required fraction of compromised secure communication links (i.e., direct keys) between non-compromised sensor nodes can be estimated as

$$\begin{aligned}
 P_e(N_c) &= \frac{(1 - p_1)p_2}{p_1 + (1 - p_1)p_2} \times \left[1 - \left(1 - \frac{c'}{M} \right)^{N_c} \right] \\
 &= \left(1 - \frac{p_1}{p} \right) \times \left[1 - \left(1 - \frac{c'}{M} \right)^{N_c} \right],
 \end{aligned}$$

where M is the key pool size.

The resilience against node capture of our scheme (ECPKS) is shown in Figure 3 with different values of M . We notice from this figure that the key pool size should be chosen larger in order to achieve higher resilience.

4.3 Communication Overhead

We observe that if two neighbor sensor nodes wish to establish a direct pairwise key during the direct key establishment phase of ECPKS, they only need to verify whether there is a pre-distributed pairwise key between them. This is done trivially by exchanging the ids of sensor nodes.

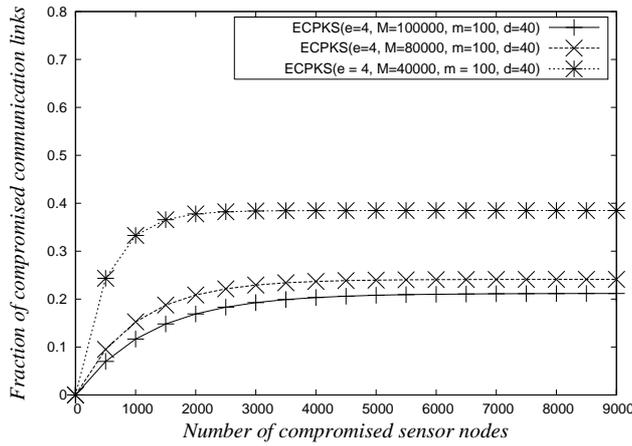


Figure 3: Resilience against node captures of ECPKS, with $n = 10000$, $d = 40$, $M = 40000$, 80000 , 100000 , $c = 200$, $m = 100$, $c' = 80$, and $e = 4$

To establish a direct pairwise key between two neighbor sensor nodes during the key prioritization phase of ECPKS, the sensor nodes only require to exchange the locations (ids) of the highest priority key units from their key rings.

As a result, the communication overhead is mainly due to transmission of the highest priority key ids from the key rings of the sensor nodes.

4.4 Computational Overhead

From the direct key establishment phase of our scheme (ECPKS), it follows that a sensor node requires one symmetric cryptographic MAC computation and another MAC verification in order to establish a secret key with a neighbor node. On the other hand, during the key prioritization phase, that node requires prioritization of the m key units in its key ring, q ($q \geq 1$) bit-wise XOR operations and two MAC operations. Since the time needed for prioritization as well as bit-wise XOR operations are negligible, the computational overhead is mainly due to four MAC operations in order to establish a secret key between two neighbor nodes during both phases.

4.5 Storage Requirement

We note that each sensor node is loaded initially with $(c+m)$ key units in its key ring. Later on unused keys from the first c key units of each node are deleted from its memory after the direct key establishment phase and again that node only keeps c' highest priority key units from the remaining m key units. Since the returned memory is used by the application part, the final storage requirement is mainly due to c' highest priority key units.

5 Comparison with Previous Schemes

In this section, we compare the performances of our scheme (ECPKS) with those for CPKS and CPPS.

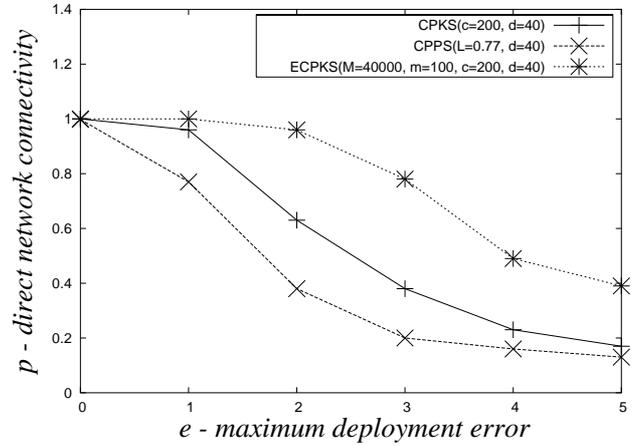


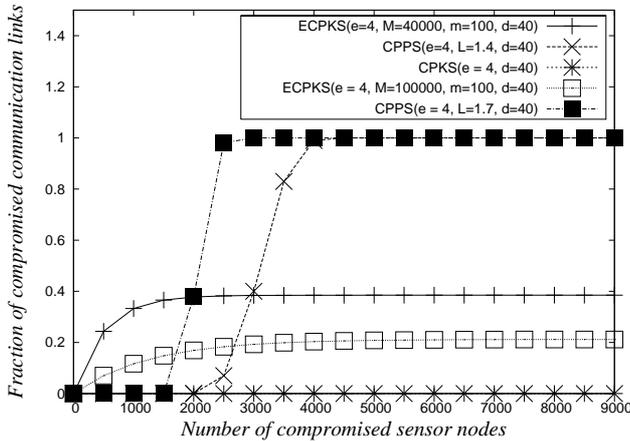
Figure 4: Direct network connectivity of ECPKS, CPKS, and CPPS. The length L of cell side in CPPS is chosen so that it is perfectly resistant to the node captures. Assume that each sensor node has available storage equivalent to 200 cryptographic keys

Figure 4 shows the comparison of network connectivity between ECPKS, CPKS, and CPPS. We assume the network size is $n = 10000$. For CPKS, we have taken $c = 200$, and $d = 40$. For CPPS, the length of cell side $L = 0.77$ is chosen so that it is perfectly resistant to the node compromise, and the value of d is taken as $d = 40$. Finally, for ECPKS, we have taken the parameter values as $M = 40000$, $m = 100$, $c' = 80$, $d = 40$, and $\rho = 30$ meters, and the size of the deployment area A is chosen so that the maximum supported network size becomes $n = \lfloor \frac{A \times (d+1)}{\pi \times \rho^2} \rfloor$. From this figure, it is clear that our proposed scheme (ECPKS) significantly improves the network connectivity than CPKS and CPPS.

Figure 5 illustrates the security against node capture of ECPKS, CPKS and CPPS. We note from this figure that CPKS provides unconditional security, because compromise of any number of nodes do not reveal any secret between non-compromised nodes in the network. The security of CPPS shows that it performs well as long as number of captured nodes in the network is small. However, when the number of captured nodes exceeds a certain threshold, CPPS leads to a large fraction of communication links between non-compromised nodes. In our scheme (ECPKS), it follows that even if an adversary captures large number of nodes in the network, ECPKS leads to only a certain fraction of communication links between non-compromised nodes. Since the key prioritization phase is applied only for d' neighbors of a sensor node where $d' = \lfloor (1 - p_1) \cdot d \rfloor$, so the security degrades only due to pairwise keys established with

Table 1: Comparison of the performances of our scheme (ECPKS) with those for the existing key pre-distribution schemes (CPKS and CPPS)

<i>Schemes</i> \Rightarrow <i>Items</i> \Downarrow	<i>CPKS</i> [14]	<i>CPPS</i> [14]	<i>ECPKS(Our scheme)</i>
Storage requirement	c key units	5 polynomial shares ($5(t+1)$ key units)	c' key units
Communication overhead	node's own id	node's home cell id	node's own id + highest priority key ids
Computational overhead	$2T_{MAC}$	evaluation of a t -degree univariate polynomial + $2T_{MAC}$	$2T_{MAC}$ (in direct key establishment) + $2T_{MAC}$ (in key prioritization)
Network connectivity	poorer if e is larger	poorer if e is larger	significantly better even if e is larger
Resilience against node capture	unconditionally secure	unconditionally secure and t collusion resistant	better than CPPS

Figure 5: Comparison of resilience against node captures between ECPKS, CPKS, and CPPS, with $n = 10000$, $e = 4$, $L = 1.4, 1.7$, $d = 40$, $M = 40000, 100000$, $m = 100$ and $c' = 80$. Assume that each sensor node has available storage equivalent to 200 cryptographic keys

those d' nodes. In CPPS, if the length L of cell side is chosen larger, CPPS leads to a larger number of sensor nodes sharing the same bivariate polynomial, which in turn degrades the security performance. CPKS is unconditionally secure against node compromise, whereas CPPS is less secure than CPKS. On the other hand, ECPKS provides better security than CPPS.

The overall comparison of the performances of our proposed scheme (ECPKS) with CPKS and CPPS is illustrated in Table 1. T_{MAC} denotes the time needed for one symmetric cryptographic MAC operation. We see that CPPS requires storage requirement for five polynomial shares which is same as storing $5(t+1)$ symmetric keys. If we assume that each sensor node has available

storage equivalent to 200 cryptographic keys, then $t = 39$ where t is the degree of a bivariate polynomial. This means that if more than t shares of that polynomial is compromised by an adversary, that polynomial is completely determined using the Lagrange's Interpolation [9]. As a result, all the keys established using that polynomial will be compromised. Thus, CPPS is unconditionally secure and t -collusion resistant. On the other hand, CPKS is unconditionally secure, whereas ECPKS is not unconditionally secure but it has significantly better security than CPPS. For network connectivity, we observe that our scheme (ECPKS) provides significantly better connectivity even for larger deployment errors than CPKS and CPPS. The communication overhead of our scheme is also comparable with CPKS and CPPS. Liu and Ning [12] pointed out that the evaluation of a t -degree polynomial requires t modular multiplications and t modular additions in a finite field $F_q = GF(q)$. Since symmetric MAC operations are efficient, the computational overheads for CPKS and ECPKS are comparable with the computational overhead for CPPS in order to establish a secret key between two neighbor nodes in the sensor network. Thus, our proposed scheme (ECPKS) is scalable, because the time needed to finish the key establishment procedure depends only on the average number of neighbor nodes rather than the total number of nodes in the network. Overall, we conclude that ECPKS is scalable and efficient in computation, communication and storage.

6 Conclusion

In this paper, we propose a scheme called the *enhanced closest pairwise keys scheme* (ECPKS) which uses simultaneously advantages of both pre-deployment as well as post-deployment knowledges of the deployed sensor nodes. ECPKS significantly improves network connec-

tivity compared to that for the existing schemes (CPKS and CPPS). ECPKS provides a better trade-off among communication overhead, computational overhead, storage overhead, network connectivity and resilience against node captures than CPKS and CPPS. In addition, it supports the addition of new sensor nodes after initial deployment and also works for any deployment topology.

Acknowledgments

The author is grateful for the constructive suggestions and comments of the anonymous reviewers which have improved the content and the presentation of this article.

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, Aug. 2002.
- [2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393-422, 2002.
- [3] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," *IEEE Symposium on Security and Privacy*, pp. 197-215, Berkeley, California, 2003.
- [4] J. Daemen and V. Rijmen, *AES Proposal: Rijndael*, AES Round 1 Technical Evaluation, CD-1: Documentation, NIST, Aug. 1998.
- [5] W. Diffie, and M.E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, pp. 644-654, 1976.
- [6] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," *ACM Conference on Computer and Communications Security (CCS'03)*, pp. 42-51, Washington DC, USA, Oct. 27-31, 2003.
- [7] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge," *23rd Conference of the IEEE Communications Society (Infocom '04)*, pp. 586-597, Hong Kong, China, Mar. 21-25, 2004.
- [8] L. Eschenauer and V. D. Gligor, "A key management scheme for distributed sensor networks," *9th ACM Conference on Computer and Communication Security*, pp. 41-47, Nov. 2002.
- [9] F. B. Hildebrand, *Introduction to Numerical Analysis*, New York: Dover, Second edition, 1974.
- [10] Crossbow Technology Inc, *Wireless Sensor Networks*, Accessed in June 2005, Online available at http://www.xbow.com/Products/Wireless_Sensor_Networks.htm.
- [11] L. Li, and J. Halpern, "Minimum-energy mobile wireless networks revisited," *Proceedings of IEEE International Conference on Communications (ICC)*, pp. 278-283, June 2001.
- [12] D. Liu, and P. Ning, "Establishing pairwise keys in distributed sensor networks," *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS)*, pp. 52-61, Washington DC, Oct. 27-31, 2003.
- [13] D. Liu, and P. Ning, "Location-based pairwise key establishments for static sensor networks," *ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN '03)*, pp. 72-82, Oct. 2003.
- [14] D. Liu and P. Ning, "Improving key pre-distribution with deployment knowledge in static sensor networks," *ACM Transactions on Sensor Networks*, vol. 1, no. 2, pp. 204-239, 2005.
- [15] D. Niculescu, and B. Nath, "Ad hoc positioning system (aps)," *Proceedings of IEEE Globecom '01*, vol. 5, pp. 2926-2931, 2001.
- [16] 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, pp. 120-126, 1978.
- [17] R. L. Rivest, "The RC5 Encryption Algorithm," *Dr. Dobbs's Journal*, vol. 20, no. 1, pp. 146-148, Jan. 1995.
- [18] W. Stallings, *Cryptography and Network Security: Principles and Practices*, Prentice Hall, 3rd edition, 2003.

Ashok Kumar Das received the M.Sc. degree in Mathematics from Indian Institute of Technology, Kharagpur 721 302, India, in 1998. He also received the M.Tech. degree in Computer Science from Indian Institute of Technology, Kharagpur 721 302, India, in 2000. He is currently towards his Ph.D. degree in Computer Science and Engineering from Indian Institute of Technology, Kharagpur 721 302, India. Prior to join in Ph.D., he worked with C-DoT (Centre for Development of Telematics), a premier telecom technology centre of Govt. of India at New Delhi, India from March 2000 to January 2004. During that period he worked there as a Research Engineer on various important projects in the fields of SS7 (Signaling System No. 7) protocol stack, GSM and GPRS. His current research interests include cryptography, information security, network security, and wireless sensor network security.