

# Protecting Neighbor Discovery Against Node Compromises in Sensor Networks

Donggang Liu  
iSec Laboratory, CSE Department  
The University of Texas at Arlington

## Abstract

*The neighborhood information has been frequently used by protocols such as routing in sensor networks. Many methods have been proposed to protect such information in hostile environments. However, these methods can only protect neighbor relations between benign nodes. A compromised node can easily circumvent them and setup false neighbor relations with sensor nodes in many places, impacting the network at a large scale. This paper presents a theoretic model for neighbor discovery in sensor networks and describes a fundamental security limitation and a generic attack against this model. The paper then proposes an efficient and localized solution based on a security property achievable during sensor deployment. This technique provides a threshold security guarantee in dealing with compromised sensor nodes. The analytical and simulation studies show that the technique is practical and effective for sensor networks.*

## 1. Introduction

In sensor networks, sensors usually interact with each other or base stations via their neighbors with which they can directly communicate. In most protocols and algorithms, a critical piece of information for every sensor node is *the list of neighbor nodes*. We assume that two nodes can directly talk to each other if they are within each other's radio range. This often implies that every pair of neighboring nodes are *physically* close to each other.

The neighborhood information has been widely used in many protocols such as routing [12], [17] and clustering [1], [16]. In routing protocols, sensor nodes need to know their neighbors to make routing decisions. In clustering algorithms, sensor nodes make their decisions based on knowledge about their neighbors. For example, a sensor node will be a cluster head if it has the smallest ID in its neighborhood [1], [2].

These protocols will not work correctly without a correct view of neighborhood information. For example, a sensor node will fail to route packets if the next hop on the routing path is not its neighbor. As another example, during cluster formation, many sensor nodes far from each other may be included in the same cluster if they do not have correct views of neighbors. In this case, communication between cluster members will become very expensive, and some data

aggregation (e.g., average in a particular area) may generate incorrect results.

Several techniques have been developed to verify whether two nodes are indeed neighbors [8]–[10], [15] by taking advantage of location information [9], [10], round trip time (RTT) [9], [10], special hardwares [8], or network topology constraints [15]. However, these schemes can only verify neighbor relations between *benign* nodes. A compromised node can easily bypass these mechanisms and do more damage to the network. For example, an attacker can replicate a compromised node and deploy many replicas in many places [14]. These replicas have all the secrets taken from the compromised node and can thus fool their nearby nodes into accepting them as a legitimate part of the network. In this way, an attacker can actively communicate with sensor nodes in many places and potentially generate a great impact on the security of many applications.

Parno et al. [14] proposed randomized and line-selected multicast schemes to detect replicas in sensor networks. Both schemes have every node digitally sign its location, producing a *location claim*. Location claims are then forwarded to some selected nodes in the network; two conflicting claims about the same node indicates a replicated node. However, they both require sensor nodes to securely discover their locations and send their location claims to witness nodes periodically. This introduces significant communication cost. A localized solution is more desirable for sensor networks.

Many techniques has been proposed recently to verify whether a node is faking its location [5], [11]. These schemes assume that adversaries only forge measurements used for localization such as received signal strength (RSS) and time of arrival (ToA). Due to the difficulties in forging consistent measurements, they can effectively detect attacks. However, they do not work effectively when there are replicated nodes since the measurements generated regarding the same replica are always consistent to each other.

In this paper, we propose to develop *localized* methods to ensure secure neighbor discovery when the attacker is able to compromise a few sensor nodes and launch various attacks based on the secrets it learned from these nodes. The goal is to prevent any compromised node from establishing neighbor relations with benign sensor nodes in many different places. The contributions are two-fold. First, we model secure neighbor discovery as a *neighbor validation* problem and describe a fundamental security limitation and a generic attack against the model. We show that the

topology information alone is not sufficient to guarantee the security of localized neighbor validation even if we can *perfectly* verify the neighbor relations between benign nodes. To develop a localized neighbor discovery with security guarantee, we propose to look for and take advantage of some security properties provided during sensor deployment. Second, we present a new localized method to protect neighbor discovery against compromised nodes. The method uses a security property that limits the attacker’s capability during sensor deployment. It offers a threshold security guarantee in dealing with compromised nodes. This method does not use the sensors’ location information, which makes it very appealing for scenarios where it is difficult to provide secure localization. Our evaluation also shows that it is practical and effective for sensor networks.

The paper is organized as follows. The next section gives the system assumptions. Section 3 models secure neighbor discovery and studies its fundamental properties. Section 4 presents the proposed secure and localized neighbor discovery protocol. Section 5 reviews related work. Section 6 concludes this paper and discusses several future directions.

## 2. System Assumptions

This paper considers static sensor networks where sensor nodes do not change their locations after deployment. The network consists of many resource-constrained sensors and a few powerful base stations. Sensor nodes are randomly scattered in a field to monitor certain events; they communicate with their neighbors to accomplish certain tasks such as data aggregation. Base stations collect/process monitoring results or act as gateways to traditional networks.

An attacker can launch a wide range of attacks against the neighbor discovery. For example, he can simply perform a denial-of-service (DoS) attack to block radio channels and prevent any sensor node in jammed areas from establishing any neighbor relation with other sensor nodes. However, in this paper, we focus on “stealthy attacks” where the attacker’s goal is to mislead the neighbor discovery. For example, he may try to fool benign sensor nodes into accepting remote sensor nodes as their neighbors. We assume that an attacker can eavesdrop, modify, forge, replay, and interrupt any network traffic. We assume that the adversary can compromise and fully control a few sensor nodes. We also assume that replicated nodes may be created and placed in the network [14].

## 3. Secure Neighbor Discovery

In hostile environments, the neighbor discovery problem can be viewed as the process to verify whether two given nodes are indeed close to each other. As mentioned, several techniques have been developed to verify whether two benign nodes are indeed neighbors [8]–[10], [15]. We refer to these methods as the *direct neighbor verification* methods, which can be used to identify the *tentative neighbors* of a sensor node and generate the *tentative neighbor relations* as defined below.

*Definition 1: (Tentative Neighbor Relation)* A tentative neighbor relation  $(u, v)$  is a neighbor relation asserted by node  $u$  using the direct verification mechanism deployed in the network.

The tentative neighbor relation  $(u, v)$  indicates that  $u$  considers  $v$  as its tentative neighbor. In this paper, we will not develop another mechanism to do direct neighbor verification. We are more interested in techniques to protect neighbor discovery given the knowledge of the tentative neighbor relations, i.e., the topology information, when there are compromised nodes. Our intuition tells us that it could be fundamentally hard to develop localized algorithms to guarantee the security of neighbor discovery if we only use topology information. If this intuition turns out to be true, we also want to know how to overcome such problem.

### 3.1. Neighbor Validation Function

We model the network topology as a directed graph and define the *neighbor validation function* for secure neighbor discovery. We assume that a direct neighbor verification mechanism has been deployed to guarantee the security of neighbor relations between benign nodes.

*Definition 2: (Tentative Network Topology)* The tentative network topology is a directed graph  $G = (V, E)$ , where  $V$  includes all sensor nodes and  $E$  includes all tentative neighbor relations.

The tentative network topology is generated by the direct neighbor verification method deployed in the network. It abstracts the field deployment of sensor nodes and represents the knowledge available for validating the neighbor relations between sensor nodes. Clearly, this topology may include incorrect neighbor relations when there are node replication attacks. Our goal is to identify those more trustworthy neighbor relations from this topology. An example of tentative network topologies is shown in Figure 1.

We use the *neighbor validation function* defined below to model secure neighbor discovery. This function is used to tell whether two given nodes are indeed neighbors based on the knowledge of tentative neighbor relations or equivalently, a subgraph of  $G$ . Note that we use the terms *subgraph* and *neighbor relations* interchangeably since each subgraph of  $G$  defines a set of neighbor relations, and each set of neighbor relations defines a subgraph of  $G$ .

*Definition 3: (Neighbor Validation Function)* Let  $I$  be set of all sensor IDs and  $\mathcal{G}$  be the set of all subgraphs of  $G$ . A function  $F: I \times I \times \mathcal{G} \rightarrow \{0, 1\}$  is said to be a *neighbor validation function* if

- For any isomorphic mapping  $f$  from  $I$  to  $I$  itself and any  $B \in \mathcal{G}$ ,  $F(u, v, B) = F(f(u), f(v), B_f)$ , where  $B_f$  is obtained by replacing every ID  $x$  in  $B$  with  $f(x)$ .

Let  $B(u)$  denote the tentative neighbor relations known by  $u$ . Given a neighbor validation function  $F$ , if  $F(u, v, B(u)) = 1$ ,  $u$  believes that  $v$  is its actual neighbor; otherwise, it will not trust the neighbor relation with  $v$ . The definition indicates that a sensor node will always make the

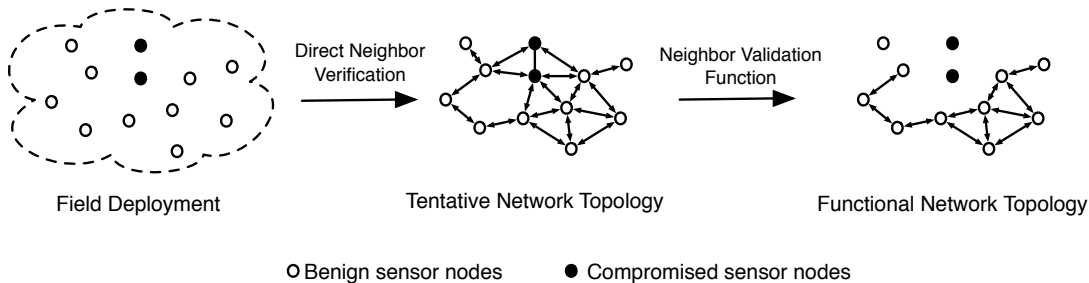


Figure 1. Tentative and Functional Network Topology

same decision about the neighbor relation with another node as long as they are connected in the same way.

Definition 3 gives a model for secure neighbor discovery that only considers tentative neighbor relations. We want to know *whether it is sufficient to only use such knowledge for neighbor validation*. Certainly, there will be other knowledge available for use, and our proposed method later does use some additional knowledge.

In practice, the neighbor validation function  $F$  may be implemented in a distributed manner. For example,  $F$  may need a large subgraph  $B$  to do the validation. Since it is undesirable for any given node to buffer a large subgraph, we may ask some nearby nodes to generate partial results on different parts of  $B$  and then collect the partial results and make the final decision. The security could be reduced since the nodes involved in the collaboration may report malicious results. This issue is out of the scope of this paper, and we consider it as one future direction.

Every sensor node will use the neighbor validation function to check every tentative neighbor node. The result of this validation will be a *functional neighbor relation*, which is defined below.

**Definition 4: (Functional Neighbor Relation)** A tentative neighbor relation  $(u, v)$  is said to be a functional neighbor relation if  $F(u, v, B(u)) = 1$  where  $B(u)$  denotes the tentative neighbor relations known to  $u$ .

Intuitively, the functional neighbor relation  $(u, v)$  indicates that  $u$  will consider  $v$  as its actual neighbor. The set of functional neighbor nodes are those actually used as neighbors in the field operations. Similar to the tentative network topology, the collection of all functional neighbor relations forms the *functional network topology*.

**Definition 5: (Functional Network Topology)** The functional network topology is a directed graph  $\tilde{G} = (V, \tilde{E})$ , where  $V$  includes all sensor nodes and  $\tilde{E}$  includes all functional neighbor relations.

A simple example of the functional network topology is shown in Figure 1. The functional network topology identifies the set of tentative neighbor relations that are more trustworthy and more likely to be the correct neighbor relations. It represents the actual topology used by the application. Hence, it is desirable to have a well-connected graph  $\tilde{G}$  in both benign or hostile situations. However, this

often makes it expensive for us to protect the neighbor discovery, as we will see later.

The functional topology  $\tilde{G}$  may include multiple, separated partitions. Ideally, we would like to have a topology where all benign nodes belong to one partition and all compromised nodes are separated from this partition. However, in practice, most of the partitions may contain both benign and compromised nodes. A partition is said to be *useful* if it can be used by the application for certain tasks. This usefulness can be defined in many ways, depending on the actual application. For example, some applications may only consider the one with the largest number of nodes; others may consider all large-enough partitions. A sensor node is said to be *non-isolated* if it belongs to a useful partition; otherwise, it is *isolated*. For example, in Figure 1, if we only consider the largest partition as useful, there are three *isolated* nodes (including the two compromised nodes).

### 3.2. Evaluation Metrics

To measure how well a given neighbor validation function is, we will study two aspects of the validation function  $F$ , the *accuracy* and the *security*. The accuracy can be simply measured by the fraction of actual neighbor relations included in the functional network topology  $\tilde{G}$ .

The security of a neighbor validation function is measured by looking at the impact of a compromised node on the neighbor discovery. In an extreme case, we will ask for *perfect safety* guarantee where any compromised node and all its replicas can only establish functional neighbor relations with a set of benign nodes that resides in a circle with radius  $R$ , where  $R$  is the maximum radio range of benign nodes in the network. In other words, the impact of a compromised node is limited in a small area. The perfect safety property is often hard, if not entirely impossible, to achieve in practice. We thus define the *d-safety* property, which says that the impact of a compromised node and all its replicas are limited in a circle with radius  $d$  ( $d \geq R$ ).

**Definition 6: (d-Safety Property)** A neighbor validation function has the *d-safety property* if for any compromised node, there exists a circle with radius  $d$  that contains all the functional neighbors of this node and its replicas.

Note that the simplest neighbor validation with security guarantee is a function that does not generate any functional neighbor relation. However, this function has the lowest

performance since the achieved accuracy is zero. A trade-off has to be made between the accuracy and the security during the design of neighbor validation functions.

In addition to the accuracy, another useful metric to measure the performance of a neighbor validation function is the size of *the minimum deployment*. This concept captures the minimum size of the subgraph needed by a neighbor validation function  $F$  to generate a functional neighbor relation. In general, the larger the size of the minimum deployment, the more expensive the validation function is.

*Definition 7: (Minimum Deployment)* Let  $F$  be a neighbor validation function. The minimum deployment  $G_{min}(F)$  is the smallest graph (i.e. has the fewest number of nodes) where there exists at least one pair of nodes  $u$  and  $v$  such that  $F(u, v, G_{min}(F)) = 1$ .

### 3.3. Security Properties

We note that a compromised node  $u$  can fool any benign node  $v$  into accepting the tentative neighbor relation  $(u, v)$ . This can be easily achieved by launching node replication attacks [14]. Thus, we always assume that the adversary can forge any tentative neighbor relation  $(u, v)$  regarding any compromised node  $u$ .

Consider a given neighbor validation function  $F$ . For security guarantee, we need to ensure its security under any possible tentative network topology. Assume the deployment field has infinite size. The following theorem shows a fundamental security limitation of any neighbor validation function that only considers topology information.

*Theorem 1:* A neighbor validation function  $F$  cannot guarantee the  $d$ -safety property for any  $d \geq R$  if  $n \geq 2m - 1$ , where  $n$  is the network size and  $m$  is the size of  $G_{min}(F)$ .

*Proof:* Assume  $n \geq 2m - 1$ . We prove the above theorem by showing a graph  $G'$  in which the  $d$ -safety property does not hold for function  $F$ .  $G'$  is constructed based on the minimum deployment  $G_{min}(F)$ . We assume that all sensor nodes are benign initially. Since we have at least  $2m - 1$  nodes, we can pick two disjoint ID sets  $A$  and  $B$  such that  $|A| = m$  and  $|B| = m - 1$ . We then use  $A$  to construct a graph  $G_A$  that is isomorphic to  $G_{min}(F)$ . We know that there exists at least one functional neighbor relation. Let  $u$  and  $w$  be the two nodes related to such relation and  $F(u, w, G_A) = 1$ .

Let  $G'_A$  be the subgraph of  $G_A$  by removing  $w$  from  $G_A$  and  $f$  be an isomorphic mapping from  $A \setminus \{w\}$  to  $B$ . We then use the nodes in  $B$  to construct another graph  $G_B$  by replacing every  $x \in A \setminus \{w\}$  in  $G'_A$  with  $f(x)$ . This graph  $G_B$  is disconnected from  $G_A$ ; the distance between  $G_A$  and  $G_B$  is made at least  $d$  in the field. In addition, we also construct a graph  $G_C$  that is disconnected from  $G_A$  and  $G_B$ . This graph includes all other sensor nodes that are not in  $A$  and  $B$ .  $G_C$  is randomly formed by only using benign sensor nodes. Finally, we have a graph  $G' = G_A \cup G_B \cup G_C$ , which can be formed by only using benign sensor nodes.

Now, let the attacker compromise  $w$  and forge the tentative neighbor relations  $G(w) = \{(w, f(x)) | x \in A \wedge (w, x) \in G_A\}$ .

These forged tentative neighbor relations will be given to node  $f(u)$  in  $G_B$ . We can see that  $G_B \cup G(w)$  is exactly the graph obtained from  $G_A$  by replacing every  $x \in A$  with  $f(x)$  except for node  $w$ . According to Definition 3, node  $f(u)$  in  $G_B$  will also consider  $w$  as its functional neighbor. Thus, the  $d$ -safety property cannot be guaranteed since the benign nodes  $u$  and  $f(u)$  are at least  $d$  away from each other.  $\square$

Theorem 1 shows a fundamental limitation for any neighbor validation function. It implies that a neighbor validation function cannot guarantee the  $d$ -safety property if it checks less than half of the nodes in the network. Note that the graph  $G'$  used in the proof includes isolated network partitions. This is realistic since the attacker can partition the network by jamming the channel between some sensor nodes.

The proof of Theorem 1 only shows the existence of a tentative topology where a neighbor validation function cannot guarantee the security of neighbor discovery. It is also interesting to study how an attacker can mount attacks against an existing sensor network.

Consider a given sensor network  $G$ . We say that this network is *extendable at  $u$*  if there is a way to deploy a new benign node  $x$  to the network such that node  $u$  will establish a functional neighbor relation with  $x$ . If the network is extendable at  $u$ , let  $R(u, x, G)$  be the set of tentative relations used by  $u$  in neighbor validation, assuming  $x$  is added to  $G$  ( $F(u, x, R(u, x, G)) = 1$ ). Note that  $x$  has not been deployed yet. However, by knowing  $F$ , we can easily predict whether a sensor network is extendable at any given node once we have the tentative network topology.

*Theorem 2:* If a neighbor validation function guarantees the  $d$ -safety property and the network  $G$  is extendable at a benign node  $u$ ,  $R(u, x, G)$  includes all non-isolated benign nodes that are more than  $d + R$  away from  $u$ , where  $x$  is the node to be deployed to make  $G$  extendable at  $u$ .

*Proof:* Assume that a non-isolated benign node  $v$  that is more than  $d + R$  away from  $u$  is not covered by  $R(u, x, G)$ . Let  $w$  be the benign functional neighbor node of  $v$ . We know that  $w$  is at least  $d$  away from  $u$ . We will then show that an attacker is able to establish functional neighbor relations with these two benign nodes  $u$  and  $w$ .

Let  $X$  be the set of tentative neighbor relations in  $R(u, x, G)$  that are related to the “to-be-deployed” sensor node  $x$  and  $Y = R(u, x, G) \setminus X$ . Let  $X_{x \rightarrow v}$  denote the set of tentative relations obtained by replacing every  $x$  with  $v$ . The attacker can simply compromise  $v$  and forge a set of tentative neighbor relations  $X_{x \rightarrow v}$ . From Definition 3, we know that  $F(u, v, X_{x \rightarrow v} \cup Y) = F(u, x, R(u, x, G)) = 1$ . Thus,  $u$  will also consider  $v$  as its functional neighbor. Hence, the compromised node  $v$  establishes functional neighbor relations with  $w$  and  $u$  that are at least  $d$  away from each other. The  $d$ -safety property cannot be guaranteed.  $\square$

The proof of Theorem 2 immediately suggests an attack against a neighbor validation function  $F$  that does not check all far away nodes. Although it may be non-trivial to figure

out  $X$  and  $Y$  used in the proof, we believe that this is not prohibitively hard for an attacker having the entire tentative network topology. A neighbor validation function  $F$  is said to be *localized* if any node who executes such function only needs to know the information about nearby sensor nodes. Theorem 2 actually indicates that it is fundamentally hard to guarantee the security of a localized neighbor validation function if only topology information is used. Indeed, a sensor node almost needs to know the entire topology to defeat node compromises.

From what we discussed so far, we can see that additional knowledge is needed to help neighbor validation. A natural and promising direction is to look for and take advantage of the security properties that can be achieved during the sensor deployment. In the next section, we will develop an efficient and localized neighbor validation protocol based on this idea. We actually use a security property that can limit the attacker’s capability in forging tentative neighbor relations through compromised and replicated sensor nodes.

#### 4. A Localized Neighbor Validation Protocol

A natural idea to guarantee the security of neighbor discovery is to develop a centralized algorithm for validating the neighborhood information. For example, we can first have a trusted base station discover the tentative network topology  $G$  and make a centralized decision for every node in the network. This idea has the potential of generating the best solution since we will have a complete view of the the network topology. However, due to the unreliable wireless link and resource constraints on sensor nodes, it is often undesirable to have a centralized neighbor discovery. We thus prefer a localized neighbor discovery with security guarantee. Unfortunately, we have already shown in Section 3 that it is impossible to achieve this by only using the topology information.

We propose to look for additional knowledge to help us validate the neighborhood information. In the following, we discuss a revised but practical adversary model where the attacker’s capability is limited during *the deployment of sensor nodes*. Specifically, we note that in many scenarios, the attacker will not be able to forge new tentative neighbor relations even if it can compromise a sensor node. For example, locating and compromising a sensor node will usually take a substantial period of time for an attacker [19]. During this time period, the node may have already finished neighbor discovery and erased the secrets to generate new tentative neighbor relations. Given this security property, it is possible to develop a localized neighbor discovery with security guarantee.

We assume that every sensor node can be trusted for a short period of time right after it is deployed, though some existing sensor nodes in the network may have already been compromised by adversaries. During this period of time, this node is able to finish certain operations such as the discovery of the initial tentative neighbor list. This assumption is

often reasonable and practical in many applications where an attacker needs to spend a certain amount of time to locate and compromise a deployed sensor node [19]. This security property can be further enhanced if we are willing to put a little more effort into the sensor deployment, for example, to make sure that there are no “visible” attackers picking up or working on the newly deployed sensor nodes in the field.

We assume that once a secret is deleted from the memory of a sensor node, it is not possible for an attacker to recover such secret even if this node is compromised later. Many ideas can be used to enhance the security of such deletion. For example, we can erase and re-write it with a random value many times. We assume that every two nodes in the field can establish a pairwise key to secure their communication. Possible techniques to achieve this include those key pre-distribution schemes developed in [3], [4], [6], [7], [13]. In the rest of this paper, we assume that the communication between any two nodes is encrypted and authenticated by their shared key, and a sequence number is used to remove replayed messages. We assume that the direct neighbor verification mechanism can always correctly verify the neighbor relation between two benign nodes.

##### 4.1. Protocol Description

Our proposed neighbor discovery protocol is based on two ideas. The first idea, which has already been mentioned before, is to limit the attacker’s capability during the deployment of sensor nodes. In particular, the deployment of sensor nodes ensures that a sensor node can finish the proposed protocol and erase the corresponding secrets before it is compromised. This makes it difficult for an attacker to forge new tentative neighbor relations even if it can compromise this node later.

The second idea of the proposed approach is to take advantage of a simple and useful property between a pair of neighboring sensor nodes, which says that two neighboring nodes usually share a large number of common neighbors. In other words, if two sensor nodes  $u$  and  $v$  are really neighbors, it is expected to see a large overlap between their tentative neighbor lists  $N(u)$  and  $N(v)$ . This provides the basis of our neighbor discovery in dealing with the impact of compromised sensor nodes. Specifically, in order to setup a functional neighbor relation between two sensor nodes  $u$  and  $v$ , they have to prove to each other that they share a certain number of neighbors.

- *Initialization:* Before deployment, a central server such as the base station generates a random key  $K$  and pre-distributes it to every sensor node. This key will be used to authenticate the neighborhood information after deployment, and will be immediately deleted after the second step, i.e., the neighbor discovery.

Every node  $u$  keeps a *binding record*  $R(u) = \{N(u), C(u)\}$ , where  $N(u)$  is the tentative neighbor list discovered by node  $u$ , and  $C(u)$  is a commitment for authenticating such neighbor list. *The record*  $R(u)$

actually binds node  $u$  to the place defined by the set of nodes in  $N(u)$ . Node  $u$  also keeps a functional neighbor list  $\bar{N}(u)$ . Initially, we have  $N(u) = \{\}$  and  $\bar{N}(u) = \{\}$ . Let  $H$  be the one way hash function. Node  $u$  will also compute a verification key  $K_u = H(K||u)$ , which will be used later to verify whether another node is a newly deployed node since a newly deployed node can always compute  $K_u$  before it deletes  $K$ . The key  $K_u$  will never be disclosed as long as node  $u$  is benign; it can only be computed by the newly deployed sensor nodes.

- *Neighbor Discovery:* After node  $u$  discovers  $N(u)$ , it generates the commitment  $C(u) = H(K||N(u)||u)$ . Node  $u$  also collects and verifies the binding record  $R(v)$  from every  $v \in N(u)$  using the pre-distributed key  $K$ . To identify functional neighbors, node  $u$  then checks every  $v \in N(u)$  to see whether  $|N(u) \cap N(v)| \geq t + 1$ , where  $t$  is a network-wise parameter. If yes,  $u$  will put  $v$  in its functional neighbor list  $\bar{N}(u)$  and also generates a relation commitment  $C(u, v) = H(K_v||u)$ . The relation commitment is used to prove that  $u$  is newly deployed (due to the fact that it computes  $K_v$ ) and considers  $v$  as its functional neighbor node. After the functional neighbor list is discovered,  $u$  will immediately delete  $K$  from its memory. Finally,  $u$  sends  $C(u, v)$  to every functional neighbor node  $v \in \bar{N}(u)$ . When node  $v$  receives such message, it can always check whether  $C(u, v)$  is valid using key  $K_v$ , which was computed by  $v$  during its initialization. If it is valid,  $v$  will put  $u$  in its functional neighbor list  $\bar{N}(v)$ .

Note that the neighbor discovery step is only performed at the newly deployed sensor nodes. For the existing sensor nodes that have already finished the neighbor discovery step, they only need to listen to the relation commitments issued to them. Once the relation commitments are successfully authenticated using their verifications keys, they need to update their functional neighbor lists. However, for any old node  $u$ , it has no way to update its binding record  $R(u) = \{N(u), C(u)\}$  by itself since the key  $K$  has been deleted from its memory. We will provide an extension later to address this problem.

Figure 2 shows a simple example of our localized neighbor validation protocol. During the initialization, the sensor node  $u$  is pre-loaded with the key  $K$  and also keeps four pieces of information, the binding record  $R(u)$ , the verification key  $K_u$ , the tentative neighbor list  $N(u)$ , and the functional neighbor list  $\bar{N}(u)$ . In neighbor discovery,  $u$  discovers five tentative neighbor nodes, and acquires their binding records  $R(1)$ ,  $R(2)$ ,  $R(3)$ ,  $R(4)$  and  $R(5)$ . Node  $u$  then finds out that only nodes 2 and 3 share more than  $t$  neighbors with itself and thus puts them in  $\bar{N}(u)$ . It will also generate the relation commitments  $C(u, 2)$  and  $C(u, 3)$ , and deletes the key  $K$  from its memory. After neighbor discovery, node  $u$  only needs to distribute its relation comments to nodes 2 and 3 to update their functional neighbor lists  $\bar{N}(2)$  and  $\bar{N}(3)$ .

## 4.2. Security

The proposed neighbor validation protocol in this section provides a threshold security guarantee in dealing with the impact of compromised sensor nodes. Specifically, we show that the impact of a compromised sensor node is limited in a small local area when there are no more than  $t$  compromised sensor nodes in the network.

*Theorem 3:* When there are no more than  $t$  compromised nodes in the network, the proposed scheme guarantees the  $2R$ -safety property, where  $R$  is the radio range of sensors.

*Proof:* We define the *original deployment point* as the location where the sensor node is first deployed. Although a sensor node could be moved to a new location after being compromised by an attacker, the original deployment point of this node will not change. Let  $d_{u,v}$  be the distance between the original deployment points of nodes  $u$  and  $v$ .

Consider a compromised sensor node  $u$ . Let  $v$  be the benign sensor node that considers  $u$  as its functional neighbor node. Since nodes  $u$  and  $v$  share at least  $t + 1$  common neighbors in their binding records, we know that at least one common neighbor is a benign sensor node. Let  $w$  be such benign sensor node, we have  $d_{v,w} \leq R$  since they are both benign nodes. Note that the attacker is not able to forge new binding records even if  $u$  is compromised. In addition, we know that node  $w$  is a benign node and is considered as a functional neighbor of node  $u$  before  $u$  is compromised. Thus,  $d_{u,w} \leq R$ . As a result, we have  $d_{u,v} \leq d_{u,w} + d_{w,v} \leq 2R$ . This indicates that the impact of node  $u$  is limited in a circle with radius  $2R$  that is centered at the original deployment of node  $u$ . Hence, the  $2R$ -safety property is guaranteed.  $\square$

From Theorem 3, we can see that the threshold  $t$  is a critical security parameter for our protocol. Obviously, we prefer a larger  $t$  to tolerate more compromised sensor nodes. However, as we will see later, a large  $t$  will reduce the accuracy (the fraction of actual neighbor relations included in the functional network topology) of our protocol. Therefore, a trade-off has to be made for configuring the threshold  $t$ . Further studies on parameter  $t$  will be given in Section 4.5.

## 4.3. Overhead

We first estimate the storage overhead. During the neighbor discovery, every node  $u$  needs to store the random key  $K$ , its own binding record  $R(u)$ , the verification key  $K_u$ , the functional neighbor list  $\bar{N}(u)$ , and the relation commitments  $\{C(u, v)|v \in \bar{N}(u)\}$ . The binding record  $R(v)$  from any tentative neighbor node  $v$  can be deleted once it is authenticated. After the neighbor discovery and the distribution of relation commitments, a sensor node only needs to remember its own binding record  $R(u)$ , the functional neighbor list  $\bar{N}(u)$ , and the verification key  $K_u$ .

For communication overhead, we note that during neighbor discovery, a sensor node  $u$  needs to identify  $N(u)$  and collect  $R(v)$  from every  $v \in N(u)$ . This only involves a

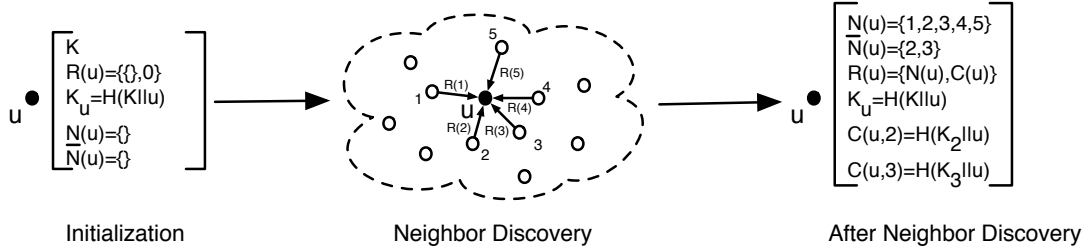


Figure 2. Localized Neighbor Validation Protocol

number of messages transmitted between neighboring sensor nodes, which we believe will not be a big problem for sensor nodes. For computation overhead, we note that the protocol only involves a few efficient one-way hash operations for authentication, which can be done efficiently in sensor networks. Hence, we believe that the proposed neighbor discovery is practical and efficient for sensor networks.

#### 4.4. Extension: Updating Binding Records

The proposed neighbor discovery protocol in this section allows the addition of new sensor nodes in the network. However, we note that once a sensor node binds itself to a set of tentative neighbor nodes, it has no way to update it later since the key  $K$  will be deleted after the initial binding. This will introduce problems when some sensor nodes run out of battery after the network is on operation for a long period of time. In particular, the number of “active” tentative neighbor nodes in the binding records of some sensor nodes will be reduced, while the newly deployed sensor nodes can only talk to those active sensor nodes. The result is that some old sensor nodes may not be able to establish new functional neighbor relations with new sensor nodes since these old sensor nodes may not have enough number of active tentative neighbor nodes in their binding records.

To deal with this problem, we propose an extension to update the binding record (i.e., add new tentative neighbor nodes in the binding record and re-generate the commitment for authenticating the updated tentative neighbor list) of an old sensor node whenever it is necessary. Specifically, we found that when a new sensor node is deployed in the network, it can use the key  $K$  to update the binding record of any node that is requesting for update. To achieve this, the requesting node needs to give evidence to prove the correctness of the new tentative neighbor relations (not included in its current binding record) discovered after it finished the neighbor discovery long time ago. The evidence will be used to convince the newly deployed sensor node to update the binding record for it. Such evidence is called the *tentative neighbor relation evidence* since it is used to prove a tentative neighbor relation.

Specifically, the tentative neighbor relation evidence from  $u$  to  $v$  will be  $E(u, v) = H(K||u||v||i)$ , where  $i$  is the *version number*, i.e., the number of times  $v$ 's binding record is updated. The version number will be included in the binding record  $R(v)$  given to  $u$  (we need to change the

format of  $R(v)$  to  $\{i, N(v), C(v)\}$ ). This evidence will be given to  $v$  via a secure channel established between  $u$  and  $v$ . It is used to prove that (1) node  $u$  does consider node  $v$  as its tentative neighbor node, and (2) such decision from  $u$  can be trusted since forging the evidence  $E(u, v)$  requires the knowledge of the key  $K$ . This also indicates that a sensor node has to remember every tentative neighbor relation evidence issued by other nodes to itself since the last update of its binding record.

During the updating process, the requesting node  $v$  will first send its old binding record  $R(v)$  as well as every relation evidence received so far to the newly deployed node  $u$ . If the version numbers included in  $R(v)$  is consistent with every relation evidence, node  $u$  starts to authenticate them. The authentication can be done easily since  $u$  has the original key  $K$ . After authentication, the new node  $u$  will update the binding record  $R(v)$  for  $v$  accordingly and send it back to  $v$ . As we mentioned, all these messages are protected by the pairwise keys shared between  $v$  and the newly deployed node  $u$ . Therefore, it is not possible for an attacker to forge or modify these messages as long as  $v$  is benign. In addition, replay attacks can be easily stopped by including a sequence number in every message.

When the node requesting for update is a malicious node, updating its binding record may reduce the security of neighbor discovery. Indeed, the compromised node can keep updating its binding record to increase its impact on the network. This is because the newly added nodes in its local area will be also used as “common neighbors” during our neighbor discovery. These common neighbors will allow the compromised node to establish functional neighbor relations with the nodes further away and thus extend its impact on the network. In practice, we will limit the number of times that a sensor node can update its binding record for security purpose. Indeed, the version number can also be used to indicate how much we can trust the binding record. Let  $m$  be the maximum number of times a binding record can be updated in the network. The following theorem shows that we still have certain degree of security guarantee.

*Theorem 4:* When there are no more than  $t$  compromised nodes in the network, the proposed extension guarantees the  $(m + 1)R$ -safety property.

*Proof:* Let  $u$  be the compromised node. Assume the binding record of  $u$  has been updated  $m'$  times. We first

show that any benign functional neighbor of  $u$  will not be more than  $(m' + 1)R$  away from  $u$ 's original deployment point. From Theorem 3, we know that the statement is true when  $m' = 1$ . Assume it is also true when  $m' = k$ . Now let us consider the case when  $m' = k + 1$ . Let node  $v$  be the benign sensor nodes that considers  $u$  as its functional neighbor node. We know that  $u$  and  $v$  share at least  $t + 1$  common neighbor nodes. Thus, at least one of them is a benign sensor node. Let  $w$  be such benign sensor node, we have  $d_{v,w} \leq R$  since they are both benign nodes. Note that node  $w$  is already in node  $u$ 's binding record. This means that node  $w$  considers  $u$  as its functional neighbor node before the  $(k + 1)$ -th update of  $R(u)$ . Thus, we have  $d_{u,w} \leq (k + 1)R$ . Note that  $u$  is not able to forge a new binding record even if it is compromised later (only the new sensor node can update the binding record for  $u$ ). We thus have  $d_{u,v} \leq d_{u,w} + d_{w,v} \leq (k + 2)R$ .

The above discussion indicates any benign functional neighbor node of the compromised sensor node  $u$  will not be more than  $(m + 1)R$  away from  $u$ 's original deployment point. Hence, the impact of node  $u$  is limited in a circle with radius  $(m + 1)R$  that is centered at  $u$ 's original deployment. The  $(m + 1)R$ -safety property is thus guaranteed.  $\square$

Providing the capability of updating binding records will increase the overhead of neighbor discovery. In particular, the newly added sensor nodes need to verify the relation evidence and update the binding records for other nodes. In addition, every node has to allocate additional memory space for buffering every relation evidence from other sensor nodes. However, these additional operations only involve the communication between neighboring sensor nodes and a few efficient hash operations. We thus believe that the proposed extension will not incur much overhead.

#### 4.5. Performance Evaluation

This subsection evaluates the proposed technique in this paper through both theoretical analysis and simulation study. Since the security and the overhead of the proposed technique have already been studied in the previous subsections, the focus of the following discussion is to evaluate the performance of the proposed neighbor discovery protocol under benign and hostile environments. According to the discussion in Section 3, we will simply use the *accuracy* and the *size of minimum deployment* to measure the performance of our protocol. We evaluate our protocol when no sensor nodes have updated their binding records yet. We consider the investigation of the effect of updating binding records on the security and the performance of neighbor discovery as one future work.

We can see that the size of minimum deployment is  $t + 3$  since two nodes need to identify at least  $t + 1$  common neighbors to setup the functional neighbor relation. Our evaluation thus focuses on the accuracy, i.e., the fraction of actual neighbor relations that is included in the functional network topology. To further simplify the analysis, we use

the fraction of actual neighbors that are included in the functional neighbor lists of benign sensor nodes to measure the accuracy in our analysis and simulation studies.

We assume that two sensor nodes can directly communicate with each other if the distance between them is less than the radio range  $R$ . We assume that the sensor nodes are randomly deployed with a uniform probability density function. We also assume that the deployment density of sensor nodes is  $D$ .

**4.5.1. Performance in Benign Situations.** Typically, in benign environments, the closer the two neighboring sensor nodes, the larger the size of the overlap of their tentative neighbor lists is. In our localized protocol, the average number of functional neighbor nodes depends on the threshold  $t$ . Let  $x = c \times R$  ( $c \leq 1$ ) be the distance between two benign tentative neighboring nodes. The average number of sensor nodes that fall in the radio range of both nodes  $u$  and  $v$  can be simply estimated by

$$N(c) = D \times R^2 \times (2 \arccos(c/2) - c\sqrt{1 - (c/2)^2}) - 2.$$

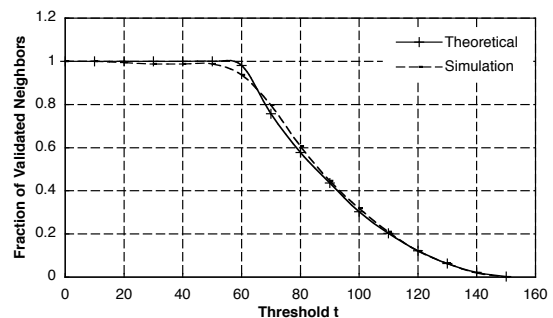


Figure 3. Fraction of actual neighbors included in the functional neighbor list of a benign node.  $R = 50$  meters.

Let  $\tau$  be the value such that  $N(\tau) \geq t + 1$ . The average number of functional neighbors can be estimated by  $D\pi\tau^2R^2 - 1$ . Thus, the fraction  $f_b$  of actual neighbors included in the functional neighbor list can be estimated by

$$f_b = \frac{D\pi\tau^2R^2 - 1}{D\pi R^2 - 1} \approx \tau^2.$$

The results of theoretical analysis and simulation study are shown in Figure 3. During the simulation, we randomly deploy 200 sensor nodes in a  $100 \times 100$  square meters field. Although this field is small when compared to practical scenarios, it is sufficient for us to evaluate our protocol in benign situations since our scheme only involves communication between neighbors and uses the information about common neighbors. Thus, we have a network with the density of one sensor node per 50 square meters. We also set the maximum radio range  $R$  to 50 meters. We focus on the sensor node located at the center of this field and obtain the simulation data from this node.

From the figure, it is clear to see that the simulation results are very close to our theoretical analysis. In addition

to this, we also find that our localized neighbor discovery protocol will not reduce the accuracy of neighbor discovery significantly for a reasonable setting of the threshold  $t$ . Note that the parameter  $t$  impacts both the resilience and the accuracy of our protocol. A small  $t$  (e.g., 30) will give us high accuracy but low security against compromised sensor nodes. A large  $t$  (e.g., 150) will give us high security against compromised sensor nodes but low accuracy since it is really uncommon to find such a large number of common neighbors between two neighbors.

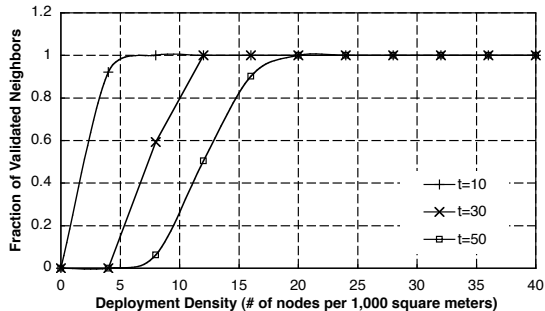


Figure 4. Fraction of actual neighbors included in the functional neighbor list of a benign node.  $R = 50$  meters.

The density of node deployment also affects the performance of our protocol. Intuitively, given the same threshold  $t$ , increasing the deployment density will increase the fraction of actual neighbors being included in the functional neighbor list of any benign node. The result shown in Figure 4 confirms this intuition. Overall, Figures 3 and 4 provide a way to configure  $t$  to trade off security with performance.

**4.5.2. Performance in Hostile Situations.** According to the security analysis in the previous subsections, we know that an attacker is not able to setup functional neighbor relations with two benign sensor nodes that are far from each other as long as there are no more than  $t$  compromised sensor nodes in the network. However, this doesn't prevent an attacker from reducing the performance of the proposed schemes.

An attacker can certainly jam the channel so that nobody can find any tentative neighbor node. We may have to use other techniques such as frequency hopping if the attacker completely jam the communication channel. Defending against the jamming attacks is out of the scope of this paper. In fact, we are more interested in the attacks that do not jam the channel but can reduce the fraction of actual neighbor nodes included in the functional neighbor list of a given sensor node  $u$ .

Fortunately, in our localized protocol, a sensor node makes its decision about another sensor node purely based on the its own tentative neighbor list and the tentative neighbor list of the other sensor node. If these two sensor nodes are both benign sensor nodes, they only need to identify  $t + 1$  commonly shared sensor nodes. Hence, the attacker has no way to reduce the number of actual benign

neighbor nodes in the functional neighbor list of any benign node  $u$  without jamming the communication channel.

**4.5.3. Comparison with Previous Techniques.** The proposed technique in this paper can be directly used to deal with the node replication attacks. Compared with the previous detection techniques proposed in [14], our scheme has a number of advantages. First, our scheme does not require the knowledge of the locations of sensor nodes in the network. This makes our approach much more appealing in situations where it is difficult to guarantee the security of location discovery. Second, our scheme can guarantee that a sensor node cannot setup neighbor relations with two benign sensor nodes that are far away from each other as long as there are no more than  $t$  compromised sensor nodes in the network. In other words, it is guaranteed that the attacker cannot duplicate a compromised sensor node at many different places and affect the network operations at a large scale. In contrast, the techniques in [14] can only detect the node replication attacks at a certain probability. Third, in our approach, a sensor node only needs to know the authenticated tentative neighbor list of another sensor node to make the decision, which only incurs a small amount of communication cost. However, the techniques in [14] involve expensive network-wide broadcast or multicast operations. Fourth, our scheme only uses a few efficient hash operations during the neighbor discovery, which is much more efficient than the public key cryptography based approaches in [14]. Finally, our scheme actually prevents an attacker from duplicating a compromised sensor node at many different places, while the previous techniques in [14] can only be used for detection, which will introduce a long delay between the time when the attacker mounts the attack and the time when it is deleted. This delay may introduce severe security problems since it allows the attacker to do a lot of damage to the sensor network during this delay.

In addition to the advantages, we must also recognize the limitation of our approach. In particular, our approach requires more deployment effort to make sure that every sensor node can be trusted for a short period of time right after the deployment. Nevertheless, as discussed before, this could be a practical direction to enhance the security of a neighbor discovery protocol given the fact that it is fundamentally hard to design a localized neighbor discovery with security guarantee. In addition, if the sensor deployment security is not guaranteed, i.e., the adversary can successfully extract the master key  $K$  in the early stage of deployment, the attacker can defeat our scheme by using this master key.

## 5. Related Work

Protecting neighbor discovery for sensor networks have been studied by many researchers [8]–[10], [15]. These techniques take advantages of location information [9], [10], round trip time (RTT) with tight time synchronization [9], [10], special hardwares (e.g., directional antennae) [8], and the constraints in the network topology [15] to make sure

that two sensor nodes are indeed neighbors. These direct verification techniques address the neighbor verification between benign nodes and are complementary to the proposed technique in this paper. In fact, the results of these schemes are used by our technique to deal with compromised sensor nodes. Another important related study to this paper is the detection of compromised sensor nodes [14], [18]. Our proposed techniques can be used to deal with this kind of attacks. These studies are complementary to each other in dealing with node compromises. They can certainly be combined together to provide better security guarantee. The comparison in the previous section have shown that our schemes have many advantages over previous techniques.

## 6. Conclusion and Future Work

In this paper, we formalize the problem of secure neighbor discovery in the presence of compromised sensor nodes and study its fundamental properties. The paper shows a number of fundamental limitations in the development of secure and localized neighbor discovery protocols. The paper then presents a novel, localized, and secure neighbor discovery protocol, which provides a threshold security guarantee against node compromises in sensor networks.

In the future, we are interested in more efficient and effective ways to do secure neighbor discovery. We will also study the performance of our technique when the direct verification mechanisms cannot guarantee the correct verification of neighbor relations between benign nodes. We also note that our protocol has to complete the neighbor discovery before we can delete the master key  $K$ . However, it may take a long time for a node to complete the neighbor discovery. An attacker will have a high chance of compromising the node and thus the master key during such time. As a result, another important direction is to study effective ideas that allow us to delete the master key  $K$  quickly without waiting for the completion of neighbor discovery.

**Acknowledgment** The author would like to thank the anonymous reviewers for their valuable comments.

## References

- [1] A.D. Amis, R. Prakash, T.H.P. Vuong, and D. T. Huynh. Max-min d-cluster formation in wireless ad hoc networks. In *Proceedings of IEEE INFOCOM 2002*, March 1999.
- [2] D.J. Baker, A. Ephremides, and J.A. Flynn. The design and simulation of a mobile radio network with distributed control. *IEEE, SAC-2(1):226–237*, 1984.
- [3] H. Chan and A. Perrig. PIKE: Peer intermediaries for key establishment in sensor networks. In *Proceedings of IEEE Infocom*, March 2005.
- [4] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy (S&P)*, pages 197–213, May 2003.
- [5] Y. Chen, W. Trappe, and R. P. Martin. Attack detection in wireless localization. In *Proceedings of IEEE INFOCOM*, 2007.
- [6] W. Du, J. Deng, Y. S. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS)*, pages 42–51, October 2003.
- [7] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, pages 41–47, November 2002.
- [8] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. In *Proceedings of the 11th Network and Distributed System Security Symposium*, pages 131–141, February 2003.
- [9] Y. Hu, A. Perrig, and D. B. Johnson. Wormhole detection in wireless ad hoc networks. Technical Report TR01-384, Department of Computer Science, Rice University, Dec 2001.
- [10] Y.C. Hu, A. Perrig, and D.B. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *Proceedings of INFOCOM*, April 2003.
- [11] J. Hwang, T. He, and Y. Kim. Detect phantom nodes in wireless sensor networks. In *Proceedings of IEEE INFOCOM*, 2007.
- [12] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of ACM MobiCom 2000*, 2000.
- [13] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS)*, pages 52–61, October 2003.
- [14] B. Parno, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. In *IEEE Symposium on Security and Privacy*, May 2005.
- [15] R. Poovendran and L. Lazos. A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks. *ACM Journal on Wireless Networks (WINET)*, 2007.
- [16] O. Younis and S. Fahmy. Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach. In *Proceedings of IEEE INFOCOM 2004*, March 2004.
- [17] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical Report UCLA/CSD-TR-01-0023, UCLA, Department of Computer Science, May 2001.
- [18] Y. Zhang, W. Liu, W. Lou, and Y. Fang. Location-based compromise-tolerant security mechanisms for wireless sensor networks. *IEEE Journal on Selected Areas in Communications, Special Issue on Security in Wireless Ad Hoc Networks*, 2006.
- [19] S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS)*, pages 62–72, October 2003.