

Resilient Cluster Formation for Sensor Networks

Donggang Liu

iSec Laboratory, CSE Department
The University of Texas at Arlington
E-mail: dliu@uta.edu

Abstract

Sensor nodes are often organized into clusters to facilitate certain network operations such as data aggregation and routing. Clustering protocols have to be protected in hostile environments. Otherwise, an attacker can easily mislead a cluster-based application by attacking the clustering protocol. This paper proposes to protect such protocol by making clustering operations accountable. Three techniques are developed for this purpose. The simple neighbor validation provides a simple yet effective way to validate a sensor's neighbors; the priority-based selection organizes clusters based on the sensor's priority of being a cluster head; and the centralized detection further enhances the security by detecting misbehaving sensor nodes. Another appealing benefit of this protocol is that a sensor node can make a clustering decision immediately once the neighborhood information (the list of neighbors) is available. This further increases the difficulty of attacking the clustering protocol. Finally, the theoretical analysis and simulation study also show that the proposed protocol is efficient and effective in dealing with malicious attacks.

1 Introduction

A typical sensor network consists of a potentially large number of low-cost, low-power, and multi-functional sensors that communicate over short distances through wireless links [1]. Sensor networks have received a lot of attention due to their attractiveness in a variety of applications such as target tracking and data acquisition. Many protocols have been developed recently to support these applications.

Sensor nodes are often organized into clusters for efficient and scalable in-network processing (e.g, data aggregation) [16], routing [20], data query [12], and broadcast [25]. Many protocols have been developed for efficient clustering [2, 3, 5, 6, 8, 20, 23, 32]. However, they can only work in benign environments. In hostile environments, an adversary can launch many attacks against these protocols. For example, the attacker may fool the sensor nodes that are far from each other into forming a cluster or hijack the role of cluster heads by forging certain information such as the node

IDs. In either case, cluster-based applications such as data aggregation may fail. This motivates the study in this paper.

However, protecting clustering protocols is quite challenging. First, the sensor nodes in the same cluster need to be physically close to each other in localized computations such as data aggregation. This is very difficult to guarantee in hostile environments. For example, an adversary can create wormholes [18] or invisible nodes [24] to fool the sensor nodes that are far from each other into forming a cluster. Second, sensor nodes are usually deployed in an unattended manner. An adversary can easily capture and compromise a few nodes [15]. When a sensor node is compromised, it can setup a neighbor relation with any node. An example of such an attack is the node replication attack [26] where the attacker duplicates the compromised node at many places. As a result, they can join many clusters even if authentication and encryption are employed.

Another challenge is the resource constraints on sensor nodes, which makes it impractical to apply well-studied but expensive mechanisms. On the other hand, the attacker may have powerful computing devices (e.g., PDAs or Laptops) and extensive knowledge about the network.

This paper presents an efficient and resilient protocol for clustering in sensor networks. The main idea is to make the clustering operations *accountable*. Specifically, we propose three techniques, *simple neighbor validation*, *priority-based selection* and *centralized detection*. The simple neighbor validation provides a simple yet effective way to validate a sensor's neighbors. The priority-based selection organizes clusters based on the sensor's priority of being a cluster head and enforces the accountability of clustering decisions made by sensor nodes. The centralized detection further enhances the security of our approach by detecting misbehaving sensor nodes using the log information generated by the priority-based selection.

These three techniques result in a resilient cluster formation scheme. On the one hand, it is very difficult for an attacker to fool the benign nodes far from each other into joining the same cluster. On the other hand, a malicious node can only impact a few benign nodes; it cannot join many clusters or recruit many benign nodes far from each other in its cluster without being detected. Another

appealing benefit is that a sensor node can make a clustering decision immediately once the neighborhood information (the list of neighbors) is available. This property makes it even harder to attack the cluster formation. In contrast, most existing protocols require a sensor node to wait for the decisions from many other nodes multiple hops away, introducing additional vulnerabilities.

This paper is organized as follows. The next section reviews existing clustering techniques. Section 3 gives the system model and design goals. Section 4 describes our technique. Section 5 presents the simulation evaluation. Section 6 reviews the related work on sensor network security. The last section concludes this paper and discusses the future work.

2 Existing Clustering Techniques

The main goal of clustering is to organize sensor nodes into clusters to facilitate certain network operations such as data aggregation. After clustering, the network is divided into disjoint clusters. Every node is covered by one cluster. Every cluster consists of a number of *cluster members* that are *physically close to each other*. One of the cluster members is used as the *cluster head* to manage the cluster.

Existing clustering methods fall into two categories, *centralized clustering* and *distributed clustering*. Centralized clustering protocols require the global network knowledge [5, 20]. Krishna et al. proposed a centralized technique where the new node runs a centralized algorithm to create new clusters and propagates the results to other nodes [20]. This method introduces substantial storage, communication and computation overheads and thus is not desirable for resource-constrained sensor nodes. Banerjee and Khuller developed a centralized clustering scheme based on a spanning tree [5]. This method requires the knowledge of network topology, which may not be always available.

Distributed clustering algorithms usually make decisions based on localized information [2, 3, 6, 8, 23, 32]. In general, distributed clustering schemes introduce less communication cost when compared with centralized schemes. The Linked Cluster Algorithm (LCA) constructs clusters using the node ID [3]. This algorithm was later improved in [10]. The Weighted Clustering Algorithm (WCA) constructs clusters based on several node properties, including the number of neighbor nodes, the transmission power, the battery-life and the node mobility [23]. The Distributed Clustering Algorithm (DCA) constructs clusters based on the weight associated with each node [6]. This weight is generic and can be defined based on different sensor applications. The Distributed and Mobility-Adaptive Clustering Algorithm (DMAC) improves DCA by adding mobility support [6]. All these schemes generate *one hop clusters* where the cluster members are no more than one hop away from the cluster head. The Max-Min d -cluster Algorithm

is able to construct clusters consisting of the nodes that are no more than d hops away from the cluster head [2]. Some other clustering protocols use probability-based approaches to elect cluster heads [4, 16, 32].

3 System Models and Design Goals

This paper considers static sensor networks where the sensor nodes do not change their locations after deployment. The network consists of the resource-constrained sensor nodes and the powerful and resourceful base stations. The sensor nodes are randomly scattered to monitor the events in the field; they need to be organized into clusters to help certain network operations such as data aggregation after deployment. The base stations are used to collect/process the monitoring results or act as gateways to the traditional networks.

Attack Models: An attacker can launch many attacks against clustering. For example, he can simply perform a denial-of-service (DoS) attack to block the wireless channel. The shared channel model [13] is largely useless here since the attacker may disrespect such model. This DoS attack is simple but common to the protocols in sensor networks; there are no effective ways to stop an attacker from mounting such attack. Therefore, We strongly believe that *a security protocol would be “good enough” if the only attack impact is equivalent to blocking the channel*. Since the security of the whole system is determined by the weakest point, and an attacker can always block the channel, the “good enough” security will not reduce the security of the whole system. We thus focus on those “stealthy attacks” whose goal is to mislead the cluster formation (e.g., recruiting the nodes far from each other in the same cluster). We assume that an attacker can eavesdrop, modify, forge, replay and interrupt network traffic. We assume that the attacker can compromise a small number of nodes. We also assume that replicated nodes may be created and placed [26].

Design Goals: An important property for a clustering protocol is that the members of the same cluster are physically close to each other. This is critical for many applications such as data aggregation [16]. For convenience, two nodes are said to be *far away* if they do not share any actual benign neighbor. For example, if the signal range can be modeled as a circle with radius R , two nodes are far away when they are at least $2R$ meters away.

Our overall goal is to make it as difficult as possible to fool the nodes far from each other into joining the same cluster. This means that *a benign node will not join another benign node’s cluster if they are far from each other, and a malicious node cannot join many clusters or recruit many benign nodes far from each other*. Note that we do not stop the attack that prevents a benign node from joining a particular cluster since this can be easily achieved by blocking the channel of the node no matter how we do.

We consider *d-hop clusters*. A node may use an intermediate node to reach the cluster head. This intermediate node is called *the uplink node*. In our technique, we have every node join the same cluster as its uplink node, which allows us to make an immediate clustering decision without the need for the decisions of other nodes. For convenience, when u uses v as its uplink node, we say “ u joins the cluster through v ” or “ v directly recruits u ”.

In an ideal situation, the security goals would be formally stated and clearly provable. Since this is difficult in the present situation, we define our goals vaguely in the following with the understanding that our quantifiable results later will show progress towards a complete solution for secure clustering.

- *Security Goal 1:* It is unlikely for a benign sensor node to select another benign sensor node (pre-determined cluster head) that is far away as its uplink node.
- *Security Goal 2:* It is difficult for a malicious node to join many clusters through benign sensor nodes.
- *Security Goal 3:* It is difficult for a malicious node to directly recruit many benign nodes far from each other.

Achieving secure clustering may reduce the performance of the protocol such as the quality of the clusters. However, in hostile environments, we believe that it is reasonable to trade off performance for security since without security, the clustering protocol will immediately fail under attacks. Although security is our number one concern, we still expect our technique to construct quality clusters (e.g., with even cluster sizes) and be efficient in terms of storage, computation and communication.

Note that a complete solution for clustering has to address cluster maintenance issues such as re-organizing clusters and re-electing cluster heads for load balancing. In this paper, however, we focus on the first step of clustering, *the initial cluster formation*. This serves as the start point towards a complete solution for secure clustering.

4 Secure Neighbor-Aware Cluster Formation

This section presents our technique for resilient cluster formation. We will first overview the protocol before presenting the scheme and analyzing its performance.

4.1 Protocol Overview

Our protocol uses the neighborhood information to form clusters. We propose three ideas to make the cluster formation *accountable* and detect malicious attacks.

Simple Neighbor Validation: Our first security goal indicates that a benign node should not select another benign node far away as its uplink node. The major issue is to verify whether two benign nodes are indeed close to each other. Many wormhole detection techniques can be used for this purpose [17, 18, 22, 28]. These techniques take advantage of

location information [18], round trip time (RTT) [18], special hardwares [17], and topology constraints [28] to make sure that two benign nodes are indeed neighbors. In this paper, *we will not spend our effort on doing the same kind of verification*. Instead, we only assume the deployment of a wormhole detection, which can verify whether two *benign* nodes are indeed neighbors with a certain accuracy.

Since current wormhole detection methods are imperfect, a sensor node will have a lot of false neighbors under large-scale wormhole attacks. Having many false neighbors often causes trouble for many protocols. With the “good enough” security in mind, we propose to simply shut down the victim nodes for a certain period of time since generating incorrect results often causes more damage to the system. It can also save the energy on the victim nodes.

A common issue of shutting down sensor nodes is that the malicious nodes may generate more negative impact on the network when there are fewer benign nodes left in a given area. Remember that an attacker can always shut down any benign node by simply blocking its communication. In this sense, shutting down benign nodes under attack for a certain period of time will not reduce the security of the system. On the other hand, we strongly believe that this issue can be addressed more effectively by the upper level application. For example, a cluster should provide correct aggregated results as long as the majority of its members are benign.

Our basic idea is to *check whether there are attacks instead of whether two nodes are neighbors*. Large-scale wormhole attacks will likely generate a large number of neighbors for the victim node. If a node notices a large number of neighbors, it can conclude that there might be malicious attacks. Obviously, there will be false positive cases. We will discuss how to reduce the false positive rate later. Note that small-scale wormhole attacks may bypass the above idea. However, we note that two neighbors usually share a large number of neighbors in benign situations. This can be used to find those more trustworthy neighbors.

Priority-Based Selection: The priority-based selection pre-determines a set of nodes that may become cluster heads. A *pre-determined cluster head* will form a cluster if it decides to be a cluster head. This idea prevents an attacker from turning a normal sensor node into a cluster head even if this node is compromised. In contrast, many existing protocols allow the attacker to compromise any node and turn it into a cluster head. The pre-determination does not have any additional overhead after deployment.

It is possible that some sensor nodes cannot find a pre-determined cluster head in their neighborhood when needed. However, as we will see in our analysis, the chance of this happening can be made very small as long as there are a reasonable number of neighbors. Considering the security advantage of pre-determining the cluster heads, we

decide to choose this option.

We construct clusters based on the *priority number* assigned to every pre-determined cluster head. This number is embedded in the node ID and is used to decide which cluster to join. A large value implies a high priority of recruiting nodes. As we will see, the priority-based selection allows a node to make the clustering decision immediately once the neighbor lists are available. This certainly makes our protocol more resilient. In contrast, most existing protocols have a sensor node wait for the decisions from many other nodes multiple hops away before making its own decision. This often becomes a critical security issue since it makes the DoS attack easier. For example, an adversary may disrupt the clustering process at a large area by blocking the channel at a few nodes. Although we focus on the “good enough” security, we still don’t want to make the attacker’s job easier than the simple channel blocking attack.

The accountability is enforced when a node joins a cluster or recruits members. We use the neighbor relations to log the clustering activities *indirectly* to enforce the accountability. More specifically, when a node wants to directly recruit another node, it has to share a large number of common neighbors with this node; when a node wants to join a cluster through a benign node, it has to collaborate with a sufficient number of neighbors. Intuitively, *the more often a malicious node behaves incorrectly, the more neighbor relations it needs to establish*. Indeed, for any node in our protocol, joining many clusters through benign nodes or directly recruiting many benign nodes far from each other will lead to the establishment of a large number of neighbor relations, providing evidences for detection.

Lightweight Centralized Detection: It is generally believed that centralized protocols are not desirable for sensor networks due to the high communication cost. However, *we strongly believe that lightweight centralized schemes are often practical in sensor networks*. The reasons are as follows. First, most applications such as data acquisition are actually centralized. Sensor networks are often capable of disseminating a few packets from every node to the base station. Our method collects the neighbor lists of sensor nodes, which are usually stable for a static sensor network. Hence, *only a few packets are needed from every node during its lifetime*. Such communication only happens when there are new nodes added. Second, a centralized scheme is usually simpler and more effective since the base station has all the necessary information and can be trusted. Third, a node can make its clustering decision independently from the detection unless there are misbehaving nodes need to be revoked.

The detection method simply checks the number of neighbors for every node. If a node has a significantly large number of neighbors, it is very likely that this node has been compromised. Again, this will generate false positive

cases, and we will discuss how to make it as low as possible. Hence, in order not to be detected, a malicious node has to reduce its malicious clustering activities. As a result, the compromise of a node can only impact a few benign nodes.

4.2 Protocol Description

Our protocol needs a pairwise key establishment protocol that can guarantee a shared key between every two nodes. Since the priority number is embedded in the node ID, an additional requirement for this protocol is the binding of the IDs with the keys to simplify the authentication of the priority number. In other words, once a node establishes a cryptographic key with another node, they both should be able to authenticate the ID as well as the priority number of the other party. One candidate scheme that meets these requirements is Blundo’s pairwise key distribution scheme [7]. We will use this scheme in our protocol.

Blundo’s scheme works as follows. Every node u is assigned with a *polynomial share* $f(u, x)$ from a t -degree symmetric polynomial $f(x, y)$, which has the property of $f(x, y) = f(y, x)$. The polynomial $f(x, y)$ is only known by the base station. To establish a pairwise key with v , u only needs to compute $f(u, v)$, which equals $f(v, u)$, the value that can also be computed by v . There is no additional communication cost involved. This simple method binds the IDs with the keys and can tolerate up to t compromised nodes [7]. For simplicity, we assume that every message between u and v is authenticated and encrypted by the key $f(u, v) = f(v, u)$. We do not consider the situation when a large number of nodes are compromised since most applications will fail in this case.

Initialization: Before deployment, the base station randomly generates a t -degree symmetric polynomial $f(x, y)$ with the property of $f(x, y) = f(y, x)$ and assigns $f(i, x)$ to every node i . In addition to this share, the base station also generates a unique key for every node to protect its communication with the base station and pre-loads two integer values m and n to every node. n is the number of priority levels, and m is the number of pre-determined cluster heads having the same priority number. Clearly, the total number of pre-determined cluster heads is $m \times n$. Every node has a unique ID between 0 and $N - 1$ for a network of N nodes. If the ID $i \leq m \times n$, it is a pre-determined cluster head with priority number $Pr(i) = 1 + \lfloor \frac{i}{m} \rfloor$. For simplicity, the priority number of a *regular node* that is not a pre-determined cluster head is considered as 0.

Simple Neighbor Validation: Let $N(u)$ be the set of u ’s neighbors before the wormhole detection. u first checks if $|N(u)|$ is larger than a threshold W . (W captures the *maximum number of a node’s neighbors in benign situations*). If yes, u will shut down itself and restart the protocol after a certain period of time, depending on the application. Otherwise, u will apply the wormhole detection to check

every node in $N(u)$. Any suspicious node will be removed from $N(u)$. In the end, $N(u)$ only includes those passed the wormhole detection.

Let $O(u, v)$ (*the overlap size*) be the number of neighbors shared by u and v . u will first calculate the overlap sizes for all the nodes in $N(u)$ and then **mark** every v in $N(u)$ if $O(u, v) < |N(u)|\beta$, where β is a system parameter, which captures *the minimum fraction of shared neighbors between two nodes in benign situations*. A marked node could be an actual neighbor but is not safe enough to be used as an uplink node.

To estimate $O(u, v)$, a simple way is to ask for $N(v)$ from v and compute $|N(u) \cap N(v)|$. However, this method allows a compromised node to forge its neighbor list and fool other nodes. We propose the following method for this problem. u first gets the neighbor list from every node in $N(u)$. For any node $v \in N(u)$, u computes the number of other neighbors that also consider v as their neighbor. This number is used as the overlap size. Specifically, $O(u, v) = |\{v' | v \in N(v') \wedge v' \in N(u)\}|$.

Priority-Based Selection: For any node u , a non-marked neighbor with a larger priority number is called the *candidate uplink node*. During the clustering, u simply uses the candidate uplink node $C(u)$ with the largest overlap size as its uplink node and joins the same cluster as $C(u)$. When there are multiple choices, we pick the one with the smallest ID to break the tie. The reason why selecting the uplink node in this way is that it can enforce *the accountability of recruiting cluster members* effectively. The log information (the neighbor relations) will be in a large number of nodes. It is very likely that many of them are benign and will report the log information correctly when needed.

Once u determines its uplink node, it has to confirm this decision for *the accountability of joining clusters*. u will be required to collaborate with its neighbors to confirm such decision so that the log information of joining clusters will also be in many nodes. Specifically, u will ask for the commitment $H(f(v, C(u)) || u)$ from every neighbor v for the decision it made, where H is an one-way hash function that is computationally infeasible to invert, and $f(v, C(u))$ is a key that can only be computed by v and $C(u)$ as we discussed before. This commitment indicates v 's acknowledgment on the uplink node $C(u)$ of u . A benign node gives *at most one* commitment for every neighbor since every node has at most one uplink node. The collection of all these commitments are used to ensure that u does select $C(u)$ as its uplink node. u then notifies $C(u)$ about its decision as well as the commitments from all its neighbors. Since $C(u)$ has the share $f(C(u), x)$, it can easily authenticate these commitments. Let $O'(C(u), u)$ be the number of verified commitments. $C(u)$ checks whether $O'(C(u), u) \geq \delta$, where δ is a system parameter, which captures *the minimum number of a node's neighbors in benign situations*. If yes,

$C(u)$ will serve as the uplink node of u ; otherwise, it will simply ignore u .

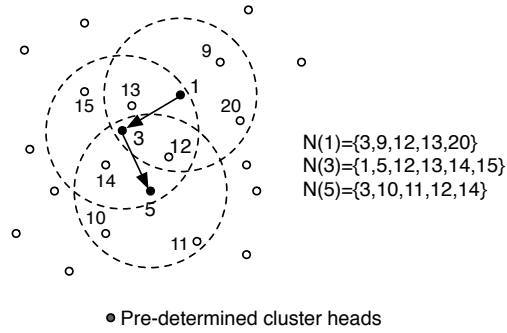


Figure 1. Priority-based selection. Assume the neighbors of 1, 3 and 5 have passed the neighbor validation. Assume $n = 3$, $m = 2$ and $\delta = 2$.

From the description, we can see that a sensor node can immediately decide which cluster to join once the neighbor lists are available. The discovery of neighbor lists only involves communications between one-hop neighbors. A sensor node may not know the ID of the cluster head when the decision is made. However, it does know how to reach the cluster head since the uplink node is identified. As shown in Figure 1, node 5 will form a cluster; node 3 will join this cluster; node 1 will join the same cluster as node 3, which is the cluster formed by node 5. Clearly, node 1 does not need to wait for node 3's decision. It simply join the same cluster as node 3.

Lightweight Centralized Detection: In the priority-based selection, the clustering activities (i.e., joining a cluster or recruiting cluster members) are accounted in the neighbor lists of sensor nodes. Hence, we can use such information to detect suspicious nodes. Specifically, every node sends its neighbor list to the base station. The base station maintains a variable $T(u)$ for every u . This variable records the the total number of other nodes that report u as their neighbor. If either $T(u)$ or $|N(u)|$ exceeds W , u will be considered as compromised. Once a node is detected to be compromised, it needs to be removed from the network. In this paper, we assume the existence of a broadcast authentication technique to do the revocation. Examples of such techniques include μ TESLA [27] and the Elliptic Curve Cryptography (ECC) based scheme [14]. μ TESLA uses symmetric key cryptography but requires time synchronization, while the ECC-based scheme uses public key cryptography but is vulnerable to the DoS attacks on signature verification. Since the broadcast authentication is only used for revoking misbehaving nodes, either one will be fine with our approach.

4.3 Security Analysis

In this subsection, we are interested in *how our technique achieves the security goals discussed in Section 3*. Since sensor nodes are often deployed to make sure that they can cover the entire field, a uniform sensor distribution is preferred in most cases. We thus assume a uniform distribution in our analysis for simplicity. We also assume two nodes can directly talk to each other if they are no more than R meters away. These assumptions may not exactly capture the real-world scenarios. However, they are often used by researchers to study the performance of various protocols in sensor networks and derive meaningful results that will be likely true too in real-world scenarios. Let N be the network size and b be the average neighbor size.

Security Goal 1: We consider the outsider attackers who do not compromise sensor nodes and the insider attackers who can compromise a few sensor nodes.

In case of outsider attackers, the security goal is achieved by the proposed neighbor validation. There are two parameters (W and β) in this method. In general, they should be configured in such a way that it is unlikely for a node to shut down itself or mark an actual neighbor in benign situations.

Lemma 1 *Let Φ be the standard normal distribution. In benign situations, the probability of a node shutting down itself or marking a neighbor is $1 - \Phi(\frac{W-b}{\sqrt{b}})\Phi((0.8-2\beta)\sqrt{b})$.*

Proof: Consider two neighbors u and v in benign situations. v will be in u 's neighbor list without being marked if (1) u has no more than W neighbors and (2) u shares at least a fraction β of neighbors with v . For the first condition, due to the uniform deployment, we can view u 's neighbor size as a binomial distribution $B(N, \frac{b}{N})$, which can be approximated by the normal distribution $N(b, b)$. For the second condition, we consider the worst case where u and v has the smallest overlap (the distance between them is R), which has the size of $\frac{2}{3}\pi R^2 - \frac{\sqrt{3}R^2}{2} \approx 0.39\pi R^2$. In this case, the probability that v shares any of u 's neighbor is 0.39. Hence, the number of shared neighbors can be viewed as the binomial distribution $B(b, 0.39)$, which can also be approximated by the normal distribution $N(0.39b, 0.2379b)$. With these two normal distributions, the probability of a node shutting down itself or marking a neighbor in benign situations is $1 - \Phi(\frac{W-b}{\sqrt{b}})\Phi((0.8-2\beta)\sqrt{b})$. \square

This lemma shows a way to set parameters properly. For example, if $b = 50$, we can set $W = 80$ and $\beta = 0.2$ so that the probability of shutting down or marking incorrectly is 0.00234. We also note that W can be often less than $2b$ for reasonable b (e.g., 50). We thus assume $W \leq 2b$.

We then study how the proposed neighbor validation achieves the security goal under outsider attacks. We assume that the detection rate and the false positive rate of detecting the wormhole between two benign nodes are r_d and r_f , respectively.

Theorem 1 *The probability of achieving the first security goal under outsider attacks is at least $1 - (1 - r_d)\Phi(\frac{(\mu-s\beta)\sqrt{s}}{\sqrt{\mu(s-\mu)}})$, where $s = b(2 - r_f - r_d)$ and $\mu = 2b(1 - r_f)(1 - r_d)$.*

Proof: Consider two benign nodes u and v that are far from each other. In the worst case, they will share $W = 2b$ neighbors before the wormhole detection. After the wormhole detection, the average number of u 's neighbors is $s = b(2 - r_f - r_d)$, and the average number of shared neighbors is at most $\mu = 2b(1 - r_f)(1 - r_d)$. The first security goal is achieved if (1) they can detect the wormhole attack or (2) they share less than $s\beta$ neighbors. For the first condition, the probability that the wormhole detection fails is $1 - r_d$. For the second condition, the number of shared neighbors can be viewed as the binomial distribution $B(s, \frac{\mu}{s})$, which can be approximated by the normal distribution $N(\mu, \sigma^2)$, where $\sigma^2 = \frac{(s-\mu)\mu}{s}$. Hence, the probability of sharing at least $s\beta$ neighbors is $\Phi(\frac{\mu-s\beta}{\sigma})$. Thus, the probability of achieving the first security goal is at least $1 - (1 - r_d)\Phi(-\frac{(\mu-s\beta)\sqrt{s}}{\sqrt{\mu(s-\mu)}})$. \square

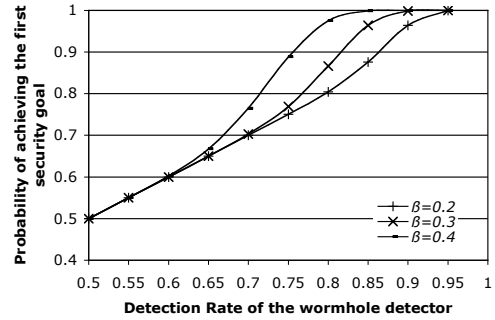


Figure 2. Probability of achieving goal 1. Assume $r_f = 0.1$ and $b = 50$

Figure 2 shows that our technique achieves the first goal with a high probability. Our neighbor validation does improve the performance of the wormhole detector. We also note that a large β leads to a high probability of achieving our goal. However, a larger β also increases the probability of an actual benign neighbor being marked as shown in Lemma 1. Nevertheless, this is often acceptable in many scenarios where the network can function well as long as the sensor nodes can find a sufficient number of neighbors to make the network well-connected.

In case of insider attackers, we always assume that the reports from sensor nodes can reach the base station for simplicity. A neighbor relation is false if the two related benign nodes are far away. We study the number N_f of false neighbor relations between benign nodes caused by insider attackers. Since any of the two benign nodes involved in a false relation has at least one neighbor relation with mali-

icious nodes, we have the upper bound $N_f \leq \frac{N_c W}{2} \leq bN_c$, where N_c is the number of compromised sensor nodes.

We are also interested in the average case where the adversary attacks random pairs of nodes that are far from each other. Consider any of these pairs. Assume the wormhole attack introduces $W = 2b$ (the worst case) neighbors. After the wormhole detection, the average number of neighbors for one node is $b(2 - r_f - r_d)$, and the average number of shared neighbors is $2b(1 - r_f)(1 - r_d)$. Thus, the attacker has to put x malicious nodes in the middle such that $(b(2 - r_f - r_d) + x)\beta \leq 2b(1 - r_f)(1 - r_d) + x$. Hence, on average, we have

$$N_f \leq \frac{N_c W}{2x} \leq \frac{N_c(1 - \beta)}{(2 - r_f - r_d)\beta - 2(1 - r_f)(1 - r_d)}$$

For example, if $\beta = 0.2$, $r_f = 0.1$ and $r_d = 0.9$, we have an average case of $N_f \approx N_c$. This shows that our technique does make it difficult for an attacker to fool a benign node into selecting another benign node that is far away as its uplink node even if there are insider attackers.

Security Goal 2: A compromised node can always bypass the wormhole detection. One way is to mount the node replication attacks [26]. We will not consider the case where a large number of nodes are compromised.

A parameter related to this security goal is δ . Generally, it should be smaller than the neighbor sizes of most benign nodes so that they can be accepted by their uplink nodes. According to the proof of Lemma 1, the probability of having less than δ benign neighbors can be easily estimated by $\Phi(\frac{\delta - b}{\sqrt{b}})$. We can thus configure δ to make this probability small enough. For example, when $b = 50$, we can set $\delta = 30$ to have a probability of 0.002339.

Clearly, a malicious node cannot convince more than W benign nodes to serve as its uplink nodes since otherwise it will be detected and revoked. In practice, our technique can have better performance when $N_c \leq \delta$.

Theorem 2 *A compromised node can join at most $\frac{W}{\delta - N_c + 1}$ clusters through benign nodes when $N_c \leq \delta$.*

Proof: For any compromised node u , if a benign pre-determined cluster head v serves as u 's uplink node, we know that u has established the neighbor relations with at least $\delta - N_c + 1$ benign nodes according to the priority-based selection. Suppose u has fooled x benign pre-determined cluster heads into serving as its uplink nodes. Since a benign node only gives at most one commitment for any given neighbor, we know that u has established the neighbor relations with at least $x(\delta - N_c + 1)$ benign nodes. If $x(\delta - N_c + 1) > W$, the proposed centralized detection will detect such misbehavior when the reports from sensor nodes can reach the base station. Hence, a compromised node can fool at most $\frac{W}{\delta - N_c + 1}$ benign pre-determined cluster heads into serving as its uplink nodes without being detected. \square

This theorem shows the upper bound on the number of clusters a compromised node can join via benign nodes.

Security Goal 3: Since only the pre-determined cluster heads can directly recruit sensor nodes, an attacker has to locate and compromise the *selected* nodes. This is the first challenge for the attacker. In addition, since the total number of neighbors for a node is limited by the threshold W , a malicious node can fool at most W benign nodes to join its cluster. Certainly, when a malicious node u recruits a benign pre-determined cluster head v , all the benign nodes recruited by v will be indirectly affected as well. However, this is generally beyond the control of the attacker.

It is usually desirable to enforce that the attack can only impact a local area. Thus, a more interesting problem is to see how many *disjoint areas* where the malicious pre-determined cluster heads can recruit cluster members. A set of areas are said to be disjoint if any two nodes in two different areas are far away. We consider the worst case where all the malicious nodes are pre-determined cluster heads.

Theorem 3 *The collusion of N_c malicious nodes can only recruit benign nodes at no more than $\frac{N_c W}{b(1 - r_f)\beta + 1}$ disjoint areas on average.*

Proof: Suppose malicious nodes recruit cluster members in k disjoint areas. Let u_i be a benign node in the i -th area that is recruited by a malicious node. Let \mathcal{N} be the set of actual benign neighbors for these k nodes. The average number of different benign neighbors discovered by these k nodes is $kb(1 - r_f)$. Since each of these k nodes joins a malicious node, the malicious nodes has to establish at least $kb(1 - r_f)\beta$ neighbor relations with the nodes in \mathcal{N} . In addition, at least one malicious node needs to setup a neighbor relation with each of these k nodes. Thus, the total number of the neighbor relations that the attacker has to establish is $kb(1 - r_f)\beta + k$. In order not to be detected, the attacker has to make sure that $k \leq \frac{N_c W}{b(1 - r_f)\beta + 1}$. \square

Note that a benign node always uses the candidate uplink node with the largest overlap size as its uplink node. Hence, we will see better performance in practice since a malicious node has to compete with the benign pre-determined cluster heads in order to be selected as the uplink node by the nodes in a given area. As a result, compromising pre-determined cluster heads can only impact the cluster formation at a small number of disjoint areas.

Other Security Properties: Since the centralized detection makes decisions based on the sensor reports, the malicious nodes can work together to revoke benign nodes from the network by forging malicious reports. An attacker can certainly make efforts to locate the nodes with the largest number of neighbors and forge reports to revoke these nodes easily. However, we are interested in average cases where an attacker wants to revoke a random

benign node. In this case, we note that the average number of neighbors discovered by a node is $b(1 - r_f)$, the attacker needs to forge $W - b(1 - r_f) + 1$ relations to revoke this node. Overall, the collusion of N_c nodes can revoke $\frac{N_c W}{W - b(1 - r_f) + 1}$ benign nodes on average.

Generally, if a benign node has more than $W - N_c$ neighbors, the attacker is able to revoke this node from the network by forging the reports. However, it is not possible for the attacker to revoke any benign node with no more than $W - N_c$ neighbors. In order to tolerate N_c compromised nodes, we can thus set a large W so that most nodes will have no more than $W - N_c$ neighbors. However, a large W will reduce the effectiveness of our protocol in achieving security. A trade-off needs to be made here to meet the practical requirements in different scenarios.

4.4 Overheads

In the proposed approach, every node needs to store a shared key with the base station and a t -degree polynomial share. This is equivalent to $t + 2$ keys. Other storage overhead includes the buffers for the neighbor lists and commitments from the neighbors. The computational overhead mainly comes from the symmetric key operations such as encryption and authentication and the polynomial evaluation during pairwise key establishment. The symmetric key operations can usually be efficiently implemented on sensor nodes [27], while the polynomial evaluation can be done efficiently by using the implementation in [21].

As for the communication overhead, the proposed technique involves the communication between neighbors and the unicast communication with the base station. The communication between neighbors are often feasible. The main concern is the unicast from every node to the base station. However, as we discussed at the beginning of this section, we believe that it is practical for every node to unicast a few reports to the base station during its lifetime.

4.5 Other Parameters

There are two other parameters in the proposed technique, the number n of priority levels and the number m of pre-determined cluster heads with the same priority number. These two parameters should be first configured to make sure that a large fraction of the nodes are covered by clusters. Thus, we expect a high probability of finding a pre-determined cluster head in the neighborhood of any node. Clearly, the probability of having no pre-determined cluster heads in a node's neighborhood can be estimated by $(1 - f)^b$, where $f = \frac{mn}{N}$. Hence, the probability of having at least one pre-determined cluster head in the neighborhood of any node can be estimated by $1 - (1 - f)^b$. To at least cover a fraction P of the nodes in the network, the fraction of pre-determined cluster heads can be easily computed by $f \geq 1 - \sqrt[b]{1 - P}$. This shows a way to configure

$n \times m$ to meet our requirements. For example, we can set $n \times m = 0.088 \times N$ so that 99% of nodes will be covered by clusters.

For any cluster formation, it is usually not desirable to have a high probability that two closely deployed nodes both claim to be the cluster heads since this may lead to a large number of small clusters and thus reduce the clustering quality. We will thus study the *collision probability*, the fraction of the actual cluster heads that have at least one actual cluster head in their neighborhoods.

Given an average of b neighbors for every node, the average number of pre-determined cluster heads in every sensor's neighborhood can be estimated by $b_p = \frac{b \times n \times m}{N}$. Consider a particular node u with a priority number i . The probability that this is the largest priority number among its neighborhood can be estimated by $(\frac{i}{n})^{b_p}$. Thus, the total number of actual cluster heads can be estimated by

$$N_{actual} = m \sum_{i=1}^n \left(\frac{i}{n}\right)^{b_p}.$$

For any pre-determined cluster head v in u 's neighborhood. The probability that v has the same priority number i can be estimated by $\frac{1}{i}$. Let $x = c \times R$ be the distance between u and v , where R is the radio range of sensor nodes. The average number of pre-determined cluster heads in the radio range of v but not u can be estimated by

$$N(x) = \frac{b_p(\pi - 2 \arccos(\frac{c}{2}) + c\sqrt{1 - \frac{c^2}{4}})}{\pi}.$$

Hence, the probability that v does not have the largest priority number among its neighbors given a certain x can be estimated by $p_i(x) = 1 - \frac{1}{i} \times (\frac{i}{n})^{N(x)}$. Due to the uniform deployment, the probability of v not having the largest priority number among its neighbors can be estimated by $p_i = \int_0^R \frac{x p(x)}{R^2} dx$. Therefore, the average number of actual cluster heads without another actual cluster head in their neighborhoods can be estimated by $N_{iso} = m \sum_{i=1}^n (\frac{i \times p_i}{n})^{b_p}$. As a result, the collision probability can be estimated by $p_{collision} = 1 - \frac{N_{iso}}{N_{actual}}$.

Figure 3 shows the collision probability when a large fraction of the nodes are covered by clusters. From the figure, we can clearly see that the collision probability decreases with n . However, a large n will generate impact on the quality of the cluster. In particular, a cluster member could be far away from its cluster head even in benign situations. Hence, we have to make trade-offs between n and the cluster quality to meet our needs. Simulation studies on practical configurations for n will be given in Section 5.

5 Simulation Evaluation

We assume sensor nodes are uniformly deployed in an 1000×1000 (square meters) field. We assume $b = 50$ and

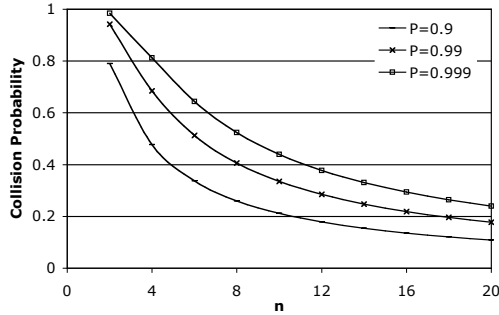


Figure 3. Collision probability under different settings of n . Assume $b = 50$.

the radio range $R = 50$ meters. This leads to a network of 6,494 nodes. We set $n \times m = 571$ so that 99% of the nodes will be covered by clusters. We set $\delta = 30$, $\beta = 0.2$, and $W = 80$ based on the results in Section 4.3. In the simulation study, we will focus on the clustering quality in benign situations and the security under attacks. We will also compare our method with existing clustering methods, in particular, the HEED protocol [32].

There are many metrics to evaluate the clustering quality. Most of them are related to the energy efficiency. Our technique is not particularly developed to optimize the energy cost. As long as the energy cost is reasonable, we believe that the cluster-based applications will work well when (1) a high percentage of nodes are covered by clusters, (2) the cluster members are close to the cluster head, and (3) most clusters have reasonable sizes. Since our technique can cover a high percentage of nodes by configuring the number of pre-determined cluster heads, we focus on the other two metrics, the hop distance to the cluster head and the distribution of the cluster size.

We configure HEED according to [32]. Specifically, we use 6 iterations and set the minimum (p_{min}) and initial ($CH_{prob} = C_{prob}$) probability of being a cluster head to 0.0005 and 0.05, respectively. We use the average distance to the neighbors to compute the cost of a given node.

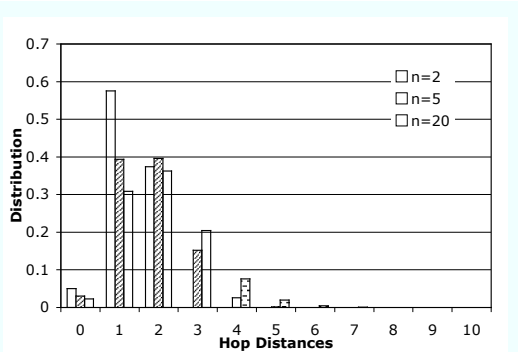


Figure 4. Hop distances to the cluster head. Assume $b = 50$, $\delta = 30$, $\beta = 0.2$, $W = 80$ and $P = 0.99$.

Hop Distances to Cluster Heads: In our technique, the hop distance from a cluster member to the cluster head is at most n in benign situations since a sensor node will always select a node with a larger priority number as its up-link node. We thus prefer a small n . However, a small n will introduce a large collision probability as shown in Figure 3, while a large collision probability will lead to a larger number of small clusters with no or only a few cluster members. Fortunately, the hop distances to the cluster head are usually much smaller than n even for a large n in practice. As shown in Figure 4, the hop distance increases much slower than n . Thus, we can usually set a large n to get a small collision probability but still have clusters with good quality (i.e., the cluster members are close to the cluster head).

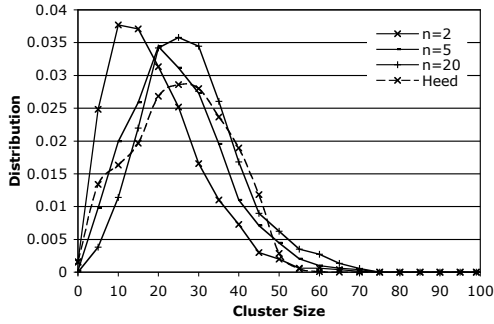
Cluster Size: We usually expect a clustering protocol to generate even clusters. In other words, the cluster sizes of most clusters are within a narrow range. Figure 5(a) shows the distribution of the cluster sizes in the network. We can see that most clusters have less than 60 nodes. We also note the fraction of small size clusters (e.g., the clusters with less than 10 members) decreases for a larger n . Together with the results in Figure 3 and Figure 4, we can find out practical configurations for n to meet different requirements.

Figure 5(a) also includes the cluster size distribution for the HEED protocol, which we will explain later.

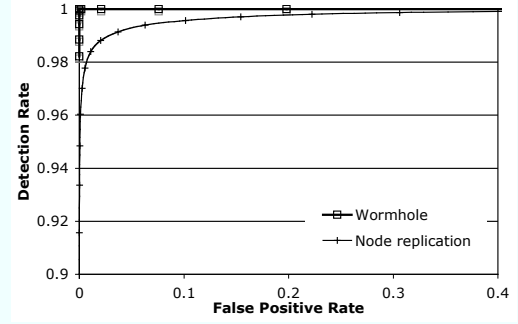
Performance under Attacks: The security of our protocol has been studied in Section 4.3. Here we use simulation to show its security under wormhole attacks [18] and node replication attacks [26]. For simplicity, the simulation will not consider the attackers who will spend a great deal of efforts to launch attacks (e.g., carefully pick the victim nodes). Nevertheless, the security of our protocol in sophisticated attacks can always be found in Section 4.3.

For the wormhole attack, the attacker creates a wormhole between a random pair of nodes that are far away. These two nodes share all their neighbors before any detection. We study the detection rate and the false positive rate. The detection rate is the probability of a node shutting down itself or marking the node at the other side of the wormhole. The false positive rate is the probability of a node shutting down itself or marking a neighbor in benign situations. We vary the threshold W and draw the ROC curve in Figure 5(b). We can see that our method can effectively detect the wormhole attack with low false positives. This is consistent with our theoretical analysis.

For the node replication attack, we assume the attacker replicates a compromised node and deploy one replicant randomly in the field. We assume the replicated node is far away from the original node. We also assume the reports from sensor nodes can reach the base station. Similarly, we are interested in the detection rate and the false positive rate. The detection rate is the probability of detecting the replicated nodes, while the false positive rate is the probability



(a) Cluster size distribution ($W=80$)



(b) ROC curves under different attacks

Figure 5. Performance of the proposed technique. Assume $b = 50$, $\delta = 30$, $\beta = 0.2$, and $P = 0.99$

of considering a node as a replicated node in benign situations. We vary the threshold W and draw the ROC curve in Figure 5(b). We can see that our technique can detect the misbehaving sensor nodes effectively. This is also consistent with our theoretical analysis.

Comparison with Previous Schemes: Though our technique is developed for resilient clustering, we still compare it with existing protocols, in particular, the HEED protocol [32], for a better understanding of its performance. First, the HEED protocol always generates one hop clusters with a stable cluster size distribution. Our protocol generates multiple hop clusters, and its cluster size distribution is comparable to that of the HEED protocol as shown in Figure 5(a). Second, the HEED protocol needs multiple (e.g., 6) iterations to terminate, and a node has to wait for the decisions from many other nodes multiple hops away. In contrast, our protocol allows a node to make a decision immediately once the neighbor lists are available. Finally, the HEED protocol will not work under attacks, while our protocol can provide resilient clustering under attacks.

6 Related Work

Section 2 has discussed the related work on sensor clustering protocols in great detail. As we mentioned, these protocols cannot work well in the presence of malicious attacks. To the best of our knowledge, the only work that deals with the malicious attacks in clustering protocols is the one proposed [30]. This scheme assumes that every node can correctly discover its neighbors, which may not be always true. This technique is complementary to ours.

A number of techniques have been developed for practical key management [9, 11, 21]. μ TESLA has been developed for broadcast authentication [27]. To build trustworthy sensor networks, researchers have studied methods to protect network services such as data aggregation [29] and routing [19]. DoS attacks against sensor networks have been thoroughly evaluated in [31]. Many techniques have been developed for wormhole detection [17, 18, 22, 28].

7 Conclusion and Future Work

This paper proposes an efficient and effective technique for resilient cluster formation, which consists of a simple neighbor validation, a priority-based selection and a centralized detection. The proposed neighbor validation is of independent interest; it can further improve the security of current wormhole detector. The analysis and simulation studies indicate that the proposed protocol is efficient and effective in dealing with malicious attacks.

There are still a number of open problems need to be addressed. In particular, it is interesting to seek effective ideas to do secure re-clustering or cluster head re-election. Additionally, we are also interested in evaluate the proposed technique under scenarios where our assumptions such as the uniform node distribution are violated.

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

References

- [1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422, 2002.
- [2] A. Amis, R. Prakash, T. Vuong, and D. T. Huynh. Max-min d-cluster formation in wireless ad hoc networks. In *Proceedings of IEEE INFOCOM 2002*, March 1999.
- [3] D. Baker, A. Ephremides, and J. Flynn. The design and simulation of a mobile radio network with distributed control. *IEEE, SAC-2*(1):226–237, 1984.
- [4] S. Bandyopadhyay and E. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *Proceedings of IEEE INFOCOM 2003*, 2003.
- [5] S. Banerjee and S. Khuller. A clustering scheme for hierarchical control in wireless networks. In *Proceedings of IEEE INFOCOM 2001*, 2001.
- [6] S. Basagni. Distributed clustering for ad hoc networks. In *Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '99)*, 1999.

- [7] C. Blundo, A. De Santis, A. Herzberg, S. Kuttan, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In *Advances in Cryptology – CRYPTO '92, LNCS 740*, pages 471–486, 1993.
- [8] H. Chan and A. Perrig. ACE: An emergent algorithm for highly uniform cluster formation. In *European Workshop on Wireless Sensor Networks (EWSN 2004)*, Jan 2004.
- [9] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Research in Security and Privacy*, pages 197–213, 2003.
- [10] A. Ephremides, J. Wieselthier, and D. Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. *IEEE Journal on Selected Areas in Communications*, 75(1):56–73, 1987.
- [11] 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*, pages 41–47, November 2002.
- [12] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of ACM MobiCom 1999*, 1999.
- [13] C. A. Gunter, S. Khanna, K. Tan, and S. Venkatesh. Dos protection for reliably authenticated broadcast. In *Proceedings of the Network and Distributed System Security (NDSS'04)*, Feb 2004.
- [14] N. Gura, A. Patel, and A. Wander. Comparing elliptic curve cryptography and rsa on 8-bit CPUs. In *Proceedings of the 2004 Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004)*, August 2004.
- [15] C. Hartung, J. Balasalle, and R. Han. Node compromise in sensor networks: The need for secure systems. Technical Report CU-CS-990-05, U. Colorado at Boulder, Jan. 2005.
- [16] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the Hawaii International Conference on System Sciences HICSS*, 2000.
- [17] 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.
- [18] Y. Hu, A. Perrig, and D. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *Proceedings of INFOCOM 2003*, April 2003.
- [19] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *Proceedings of 1st IEEE International Workshop on Sensor Network Protocols and Applications*, May 2003.
- [20] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster-based approach for routing in dynamic networks. *SIGCOMM Computer Communication Review*, 27(2), 1997.
- [21] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, pages 52–61, October 2003.
- [22] D. Liu, P. Ning, and W. Du. Detecting malicious beacon nodes for secure location discovery in wireless sensor networks. In *Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS '05)*, June 2005.
- [23] S. D. M. Chatterjee and D. Turgut. Wca: A weighted clustering algorithm for mobile ad hoc networks. *Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks)*, 5(2):193–204, 2002.
- [24] J. Marshall. An analysis of srp for mobile ad hoc networks. In *Proceedings of the 2002 International Multiconference in Computer Science*, August 2002.
- [25] E. Pagani and G. Rossi. Reliable broadcast in mobile multihop packet networks. In *Proceedings of MobiCom' 97*, 1997.
- [26] 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.
- [27] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar. SPINS: Security protocols for sensor networks. In *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks*, July 2001.
- [28] 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) (to appear)*, 2006.
- [29] B. Przydatek, D. Song, and A. Perrig. SIA: Secure information aggregation in sensor networks. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys '03)*, Nov 2003.
- [30] K. Sun, P. Peng, P. Ning, and C. Wang. Secure distributed cluster formation in wireless sensor networks. In *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC'06)*, 2006.
- [31] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, 2002.
- [32] 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.