8-16-2007

Secure Neighbor Discovery in Wireless Sensor Networks

Saurabh Bagchi
Purdue University, sbagchi@purdue.edu

Srikanth Hariharan
Purdue University, srikanth@purdue.edu

Ness Shroff
Purdue University, shroff@purdue.edu

Secure Neighbor Discovery in Wireless Sensor Networks

Srikanth Hariharan, Ness B. Shroff and Saurabh Bagchi

School of Electrical and Computer Engineering, Purdue University

Email: {srikanth, shroff, sbagchi}@purdue.edu

Abstract. Wireless Sensor Networks are increasingly being used for data monitoring in commercial, industrial, and military applications. Security is of great concern from many different viewpoints: ensuring that sensitive data does not fall into wrong hands; ensuring that the received data has not been doctored; and ensuring that the network is resilient to denial of service attacks. We study the fundamental problem of Secure Neighbor Discovery problem, which is critical to protecting the network against a number of different forms of attacks. Sensor networks, deployed in hazardous environment, are exposed to a variety of attacks like eavesdropping, message tampering, selective forwarding, wormhole and sybil attacks. Attacks against the data traffic can be addressed using cryptographic techniques. We first present an efficient and scalable key-distribution protocol which is completely secure in the absence of colluding malicious nodes. Secure neighbor discovery can help to defend against a majority of the attacks against control traffic. We consider a static network and propose a secure one-hop neighbor discovery protocol. We show by analysis that this protocol effectively prevents two non-neighboring nodes from becoming neighbors even when both the nodes have been compromised by the adversary. We then extend this protocol so that it works even when nodes are incrementally deployed in the network. We also briefly study how this protocol could be modified for mobile sensor networks. Finally, we compare our protocol with existing neighbor discovery protocols and analyze the advantages and disadvantages of using these protocols.

1 Introduction

1.1 Wireless Sensor Networks

A wireless sensor network is a wireless network made of numerous small sensor nodes. The sensor nodes are self-contained units consisting of a battery, radio and sensors and a processor with minimal computation power. Thus, they are resource starved devices with a minimal amount of memory, energy and computation power. These nodes perform a variety of functions including sensing, communication and computation. They may be static nodes which stay in a fixed position throughout their lifetime or mobile nodes which may move to various locations depending on the function that they need to perform. Wireless sensor networks are a particular class of wireless ad-hoc networks which do not need any fixed routing units or base stations in order to communicate. Each sensor node can communicate with nodes that are within its communication range. We call nodes that are within the communication range of a sensor node as its one-hop neighbors, or just neighbors. Intermediate nodes perform the routing operation when a sensor node wants to communicate with a node that is not within its communication range and the communication is usually by wireless RF links that have a low bandwidth.

1.1.1 Deployment

Sensor networks are used in commercial and industrial applications to monitor data that would be difficult or expensive to monitor otherwise. They could be deployed in wilderness areas, where they would remain for many years without the need to recharge or replace their power supplies. Since the environment that they monitor is generally hostile, it is usually not possible to deploy each node in a known location. Sensor nodes might therefore be scattered from a plane onto the region that they would be monitoring. Consequently, a sensor node, upon deployment, does not have knowledge of the nodes that are its neighbors.

1.1.2 Communication

Like wireless ad-hoc networks, sensor networks have two modes of communication.

1. Local Broadcast (One to Many)

2. Node to Node (One to One)

When a sensor node sends a packet by local broadcast, all its neighbors receive the packet. In the other case, the sensor node can send the packet to a specific node alone.

1.2 Security Attacks on Wireless Sensor Networks

Due to the open nature of communication and the hostile environments in which sensor nodes are deployed, security becomes a critical concern in sensor networks. An adversary can eavesdrop on packets, tamper messages, spoof identity and can also unleash a variety of routing (blackhole, wormhole) and physical layer attacks. Nodes can be compromised and compromised nodes might collude.

To protect against eavesdropping, message tampering and identity spoofing, a variety of cryptographic protocols for sensor networks have been proposed that address the encryption and authentication issues. Routing and physical layer attacks are still a major concern and [12], [29], [10], [11], [13], [14], [15] suggest measures to protect against these attacks.

1.3 Problem Statement

We focus on the problem of secure neighbor discovery in wireless sensor networks. Recent work [13], [15], [14], [16], [35], among others, have assumed that the time taken to compromise a sensor node is greater than the time required for neighbor discovery. Since the time taken to compromise a sensor node and the time required for neighbor discovery are both expected to be of the order of seconds, there is a chance that a very small fraction of the nodes are compromised before they perform neighbor discovery.

Neighbor discovery, if not correctly performed, can lead to the launch of serious security attacks against the network. For instance, an adversary who wants to unleash a wormhole or a sinkhole attack [12] will want to make his neighbors believe that he lies on the best routing path. Once he succeeds in this operation, he gains control over the routing path and can selectively forward or drop packets, tunnel them to another adversary, etc.

We propose a secure neighbor discovery protocol for WSNs. One of the fundamental requirements in any security protocol is an efficient way of key distribution and management. Therefore, we first present an implementation of an existing key pre-distribution scheme [2]

which is memory efficient, scalable, does not incur any communication overhead and is secure as long as there are no colluding malicious nodes in the network. We use this scheme for initial secure authentication between any pair of nodes. This scheme is very efficient as long as there are no colluding malicious nodes in the WSN.

We, then, propose a secure one-hop neighbor discovery protocol for static sensor networks when all the nodes are deployed initially. Neighbor discovery protocols are vulnerable to a variety of attacks that could either prevent two neighboring nodes from becoming neighbors or could make two non-neighboring nodes to believe that they are neighbors. Our protocol focusses on the latter issue. We show how the adversary can use a specific form of the wormhole attack and make two nodes that are not within the communication range of each other to believe that they are neighbors and show how our protocol effectively counters such an attack. We do not protect against brute force denial of service attacks, such as physical destruction of nodes or physical layer jamming.

We then consider the case in which sensor nodes are incrementally added to the network. Sensor nodes are prone to natural failure. Also, there might be a need to deploy more nodes when the sensing operation needs more redundancy for better accuracy. Incremental addition might also be necessary when a compromised node has been detected and removed from the network. So we study how to perform neighbor discovery in such a scenario. We show how our protocol could be easily extended so that it could be used even when nodes are incrementally deployed in the WSN.

We further study briefly about the problem of neighbor discovery in mobile networks and how our protocol could be modified to handle mobility.

Finally, we compare our protocol with existing neighbor discovery protocols. We analyze the advantages and disadvantages of using these protocols. This provides us an insight into designing better protocols to solve this problem.

1.4 Contributions

- Modified an existing key pre-distribution scheme and presented a scalable, secure key management scheme that incurs significantly lesser communication overhead compared to other existing protocols.

- Proposed a secure neighbor discovery protocol for static sensor networks which effectively prevents two non-neighboring nodes from convincing themselves as well as their other actual neighbors that they are neighbors.
- Extended this protocol for incremental node deployment and studied briefly about using this protocol for mobile networks.
- Compared between various neighbor discovery protocols and suggested factors that should be taken into consideration in order to design better protocols.

1.5 Outline

The rest of this paper is organized as follows. Section 2 talks about the key establishment protocol. Section 3 describes the secure neighbor discovery protocol for static sensor networks. Section 4 explains the extension of this protocol for incremental node deployment and for handling mobility. Further, Section 4 compares between existing neighbor discovery protocols and analyzes the advantages and disadvantages of using such protocols. Section 5 presents the related work in this field. Finally, Section 6 concludes the paper and provides directions for future research.

2 Key Distribution

2.1 Key Pre-distribution - An Introduction

Key establishment in sensor networks is a challenging problem because of the resource constrained nature of these networks. Assymetric key cryptosystems have been generally agreed in the literature [5], [7], [4], [28] to be computation intensive and therefore unsuitable for sensor networks. A lot of symmetric key cryptographic protocols have therefore been analyzed. The primary goals that an ideal symmetric key cryptosystem for sensor networks must achieve have been summarized below:

- Secure communication between any two nodes.
- Memory-scalable: By memory scalability, we mean that when the number of nodes in

the networks increase by an order of magnitude, the number of keys that each node needs to store should increase gradually.

- Low communication and bandwidth overhead.
- Energy-aware: Since communication consumes the maximum energy in sensor nodes, the sensor nodes are expected to sleep during a majority of the time.
- A graceful degradation in performance when nodes get compromised.

A simple and naive solution that ensures secure communication between any pair of nodes would be to have a pair-wise key between any two nodes. But such an approach is obviously not scalable. At the other extreme, we might have a symmetric key management protocol that relies on a common shared secret key between all the nodes in the network leading to a highly insecure deployment. The additional requirement to minimize communication overhead makes most of the proposed purely symmetric algorithms impractical for WSNs.

In [2], Blom proposes a key pre-distribution scheme that allows any pair of nodes to find a pair-wise key between them. Compared to the $(N-1)$ pair-wise key pre-distribution scheme, Blom's scheme uses only $\delta + 1$ memory spaces with δ much smaller than N. The tradeoff is that, unlike the $(N-1)$ pair-wise key scheme, Blom's schmee is not perfectly resilient against node capture. If $\delta + 1$ nodes are compromised and they collude, all pair-wise keys of the entire network are compromised. But, as δ increases, the computational and storage overhead increase. [3], [6], [20] extend Blom's work to provide higher scalability and a larger number of nodes to be compromised in order to expose the entire network.

A different flavor of protocols [16], [28] enable secure communication between any pair of nodes irrespective of the number of nodes compromised but they require each node to communicate with the base station initially, thus incurring a large communication overhead.

Since the focus of our problem is secure neighbor discovery, our key pre-distribution protocol uses an implementation of Blom's scheme for initial authentication and neighbor discovery. In the absence of colluding nodes, the protocol guarantees that the communication between any two non-compromised nodes is secure irrespective of the number of nodes compromised in the network.

Since this protocol becomes increasingly ineffective in the presence of colluding nodes, the keys cannot be used for communication when malicious nodes begin to collude. Therefore, these keys can only be treated as temporary keys and should be deleted after initial usage. New keys could be established with the help of the base station. The sections that follow talk about the system model and the details of our key distribution protocol.

2.2 System Model and Assumptions

The WSN is deployed within a huge field which has been pre-determined.

2.2.1 Assumptions

Links between sensor nodes are assumed to be bi-directional. By bi-directional links, we mean that two nodes are defined to have a link between them iff they can hear each other's transmission.

2.3 Attack Model

An attacker can be either an external node that does not know the cryptographic keys, or an insider node, that possesses the keys. An insider node may be created by compromising a legitimate node. All these malicious nodes can collude among themselves. Any malicious node can eavesdrop on the traffic, tamper with messages, indulge in identity spoofing attacks, or tunnel network traffic from one location of the network to a colluding node in another location (wormhole attack). They can also buffer messages sent by a legitimate node and read its messages when aone of its links is compromised.

2.4 The Key Pre-Distribution Protocol

2.4.1 Group key establishment

Let the number of sensor nodes that are going to be deployed initially be N. We arrange these nodes in a virtual square grid ($\sqrt{N} \times \sqrt{N}$). For simplicity, lets assume that N is a perfect square. The elements in the grid are referred by ij, $1 \leq i \leq \sqrt{N}$ and $1 \leq j \leq \sqrt{N}$, where i denotes the row and j denotes the column. We call ij as the *Node ID* of the sensor node in row i and

column j. Each sensor node has a pseudo-random function F that takes three keys as input and returns a unique random key as its output.

An example of a 7×7 virtual grid is shown below.

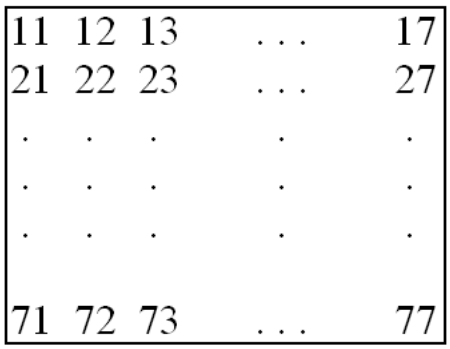


Figure 1: Example of a 7×7 virtual grid.

We now divide the sensor nodes into three types of groups.

1. **The Row Group :** Sensor nodes in each row of the virtual grid share a common key with other nodes in its row and a unique pair-wise key with sensor nodes in every other row. For example, consider row i in the virtual grid. Each node $i1, i2, ..., i\sqrt{N}$ share a common key which we shall denote by R_{ii}. Each node in row i also shares a common key with each node in row j . We shall call the key that nodes in row i share with nodes in row j as R_{ij}.

2. **The Column Group :** This is similar to the row groups. Sensor nodes in each column of the virtual grid share a common key with other nodes in its column and a unique pair-wise key with sensor nodes in every other column. For example, in column i, the common key shared between $1i, 2i, ..., \sqrt{N}i$ is denoted by C_{ii} and the key shared between column i and column j shall be denoted by C_{ij}.

3. **The Diagonal Group :** Apart from the row and column groups, sensor nodes in each diagonal of the virtual grid share a common key with other nodes which lie on the same diagonal and a unique pair-wise key with nodes in every other diagonal. We now explain the numbering scheme for the diagonal. A node ij lies on the diagonal numbered by $(j-i)$. The common key shared between nodes that lie on the same diagonal is denoted

by $D_{(j-i)(j-i)}$. Consider nodes i_1j_1 and i_2j_2 lying on different diagonals. The key shared between them is denoted by $D_{(j_1-i_1)(j_2-i_2)}$.

Since all the keys are symmetric, $R_{ij} = R_{ji}$, $C_{ij} = C_{ji}$ and $D_{ij} = D_{ji}$.

2.4.2 Deriving pair-wise keys

Using the setup above, each node can derive the unique key that it shares with any other node by knowing each other's *Node ID*. Let us consider two nodes i_1j_1 and i_2j_2 wanting to derive the key that they share. If the derived key is denoted by K, then

$$K = F(R_{i_1i_2}, C_{j_1j_2}, D_{(j_1-i_1)(j_2-i_2)}) \tag{1}$$

Let us study a simple example. Consider a network with 25 nodes, arranged in a $5 * 5$ virtual grid.

We will now see how keys can be established between different pairs of nodes.

1. Consider two nodes in the same row, say 21 and 23. The key shared between these two nodes is given by $F(R_{22}, C_{13}, D_{-11})$.

2. Consider two nodes in the same column, for instance, 12 and 22. The key shared between them is given by $F(R_{12}, C_{22}, D_{10})$.

3. Next consider two nodes in the same diagonal, say 21 and 32. The key shared between them is given by $F(R_{23}, C_{12}, D_{-1-1})$.

4. Finally consider two arbitrary nodes, say 11 and 23. The key shared between them is given by $F(R_{12}, C_{13}, D_{01})$.

2.4.3 Storage overhead

We now perform an analysis on the number of keys that need to be stored by each node and by the network, on the whole. We show that this scheme requires only $O(\sqrt{N})$ keys to be stored at each node and $O(N)$ keys to be stored by the entire network, thus improving the scalability over existing key distribution protocols.

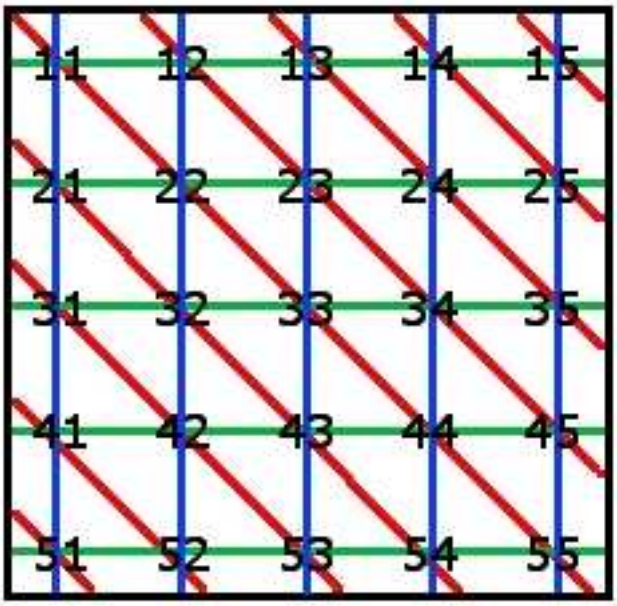


Figure 2: Example of key establishment in a 5×5 virtual grid.

Memory requirement at each node Since the virtual grid contains $\sqrt{N}$ rows, $\sqrt{N}$ columns and $2\sqrt{N}-1$ diagonals, each node has to store $\sqrt{N}$ Row Group keys, $\sqrt{N}$ Column Group keys and $2\sqrt{N}-1$ Diagonal Group keys. Thus, each node needs to store $4\sqrt{N}-1$ keys. Thus, this scheme requires $O(\sqrt{N})$ keys to be stored at each node.

Memory requirement for the whole network Let T_{RGK}, T_{CGK} and T_{DGK} represent the total number of Row Group keys, Column Group keys and Diagonal Group keys respectively, stored by the network. We then have,

$$T_{RGK} = 1 + 2 + ... + \sqrt{N} \tag{2}$$

$$T_{CGK} = 1 + 2 + ... + \sqrt{N} \tag{3}$$

$$T_{DGK} = 1 + 2 + ... + 2\sqrt{N} - 1 \tag{4}$$

Thus if T_N denotes the total number of keys stored by the network, we have

$$\begin{aligned} T_N &= T_{RGK} + T_{CGK} + T_{DGK} \\ &= \sqrt{N}(\sqrt{N}+1) + \sqrt{N}(2\sqrt{N}-1) \end{aligned}$$

$$= 3N \tag{5}$$

Thus, the network needs to store $O(N)$ keys. This is a significant improvement compared to the $O(N^2)$ keys that would be required by a protocol that stores a distinct key between any pair of nodes in the network.

2.5 Security Analysis

It has been proved that in such a pre-distribution scheme, the presence of t compromised colluding malicious nodes can expose the entire network ([3], [8]).

Proposition:

The communication between any two nodes, N_1 and N_2, is secure irrespective of the number of nodes compromised as long as all the following conditions hold:

1. Neither of the two nodes, N_1 and N_2, is compromised.
2. There are no colluding malicious nodes in the network.

Proof:

If either of N_1 or N_2 is compromised, the communication between N_1 and N_2 is exposed. Therefore, condition 1 is necessary.

Let P be the set of compromised nodes in the network. $N_1 \notin P$ and $N_2 \notin P$. Let R_{12}, C_{12} and D_{12} denote the row, column and diagonal group keys, respectively, that N_1 shares with N_2. To prove that condition 2 must also hold along with condition 1:

None of the compromised nodes collude.

For any $M \in P$,

1. If M is in the same row as N_1 or N_2, M knows R_{12};
2. If M is in the same column as N_1 or N_2, M knows C_{12};
3. If M is in the same diagonal as N_1 or N_2, M knows D_{12};
4. Otherwise M does not know any of the keys that N_1 and N_2 share.

Since no nodes collude, M can obtain access to at the most two keys out of R_{12}, C_{12} and D_{12}. Therefore, M cannot obtain the shared key between N_1 and N_2.

This completes the proof.

We now present a simple example to show that in the presence of three colluding malicious nodes, communication between a lot of nodes is exposed.

Consider a network with 16 nodes arranged in a $4*4$ virtual grid as shown in Figure 3. We will denote the nodes by N_{ij}, where $1 \leq i \leq 4$ and $1 \leq j \leq 4$, for convenience. Suppose N_{13}, N_{21} and N_{34} are compromised and they also collude with each other. Then, apart from the communication between any node and any of these compromised nodes, the communication between the following pairs of nodes is also exposed: N_{11} and N_{12}; N_{11} and N_{23}; N_{11} and N_{24}; N_{11} and N_{32}; N_{11} and N_{43}; N_{12} and N_{14}; N_{12} and N_{23}; N_{12} and N_{24}; N_{12} and N_{31}; N_{12} and N_{33}; N_{12} and N_{41}; N_{12} and N_{43}; N_{12} and N_{44}; N_{14} and N_{23}; N_{14} and N_{24}; N_{14} and N_{32}; N_{14} and N_{43}; N_{22} and N_{23}; N_{22} and N_{24}; N_{22} and N_{32}; N_{22} and N_{43}; N_{23} and N_{24}; N_{23} and N_{31}; N_{23} and N_{32}; N_{23} and N_{33}; N_{23} and N_{41}; N_{23} and N_{42}; N_{23} and N_{43}; N_{23} and N_{44}; N_{24} and N_{31}; N_{24} and N_{32}; N_{24} and N_{33}; N_{24} and N_{41}; N_{24} and N_{42}; N_{24} and N_{43}; N_{24} and N_{44}; N_{31} and N_{32}; N_{31} and N_{43}; N_{32} and N_{33}; N_{32} and N_{41}; N_{32} and N_{43}; N_{32} and N_{44}; N_{33} and N_{43}.

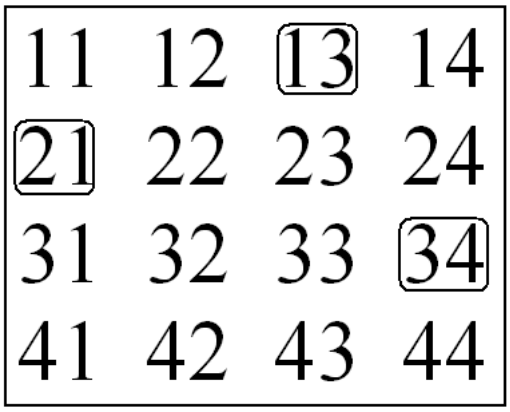


Figure 3: Security analysis in a 4×4 virtual grid. The boxed nodes are the malicious colluding nodes.

Thus, a huge fraction of communication can become exposed in the presence of three colluding malicious nodes.

2.6 Incremental Deployment

In this section, we briefly describe how this matrix based key pre-distribution scheme could be made to handle incremental node deployment. We assume that the maximum number of nodes that would be deployed in the WSN is known and analyze the number of additional keys that need to be pre-distributed to each node so that it can securely establish communication with incrementally deployed nodes.

2.6.1 The technique

We assume that each node that has been already deployed knows the ID of the last node that was deployed in the WSN. Nodes are deployed in the WSN in the following order: (i, i), $(i-1, i)$, $(i, i-1)$, $(i-2, i)$, $(i, i-2)$, ... , $(1, i)$, $(i, 1)$, $(i+1, i+1)$, ... and so on.

Let the maximum number of nodes that would be deployed in the network be N_{MAX} and the number of nodes initially deployed be N. Each node then stores $4\sqrt{N_{MAX}} - 1$ keys while it uses only $4\sqrt{N} - 1$ keys. Therefore, for $N_{MAX} - N$ additional nodes to be deployed, each sensor node has to store $4(\sqrt{N_{MAX}} - \sqrt{N})$ additional keys.

3 Secure Neighbor Discovery in Static Sensor Networks

This section suggests a protocol for secure one-hop neighbor discovery in WSNs in which the sensor nodes are static. One of the important characteristics of WSNs is that they are self-configuring, i.e., a large number of wireless nodes organize themselves to efficiently perform the tasks required by the application after they have been deployed. One-hop neighbors of a node are those which are within the radio communication range of the node. By secure neighbor discovery, we mean that *for any node in the WSN, no node that is not within its one-hop communication range can become its neighbor.* Malicious nodes that are within the communication range might not respond to *Hello* packets sent by certain nodes. If a node does not respond, it is only isolating itself and therefore cannot launch security attacks that are more devastating than when it responds to *Hello* packets. Discovery of one-hop neighbors is essential for a variety of applications which we will study in the next section. We then describe our neighbor discovery protocol and analyze it for static sensor networks. We compare our

protocol with a protocol that uses directional antenna for neighbor discovery (proposed by Hu and Evans), [9], and show that our protocol performs significantly better in a lot of aspects.

3.1 Importance of Secure Neighbor Discovery

Knowledge of one-hop neighbors is essential for almost every routing protocol, MAC protocols and several other topology-control algorithms such as construction of minimum-energy spanning trees. Neighbor disovery is, therefore, a crucial first step in the process of self-organization of WSNs. Recently, neighbor discovery has also played a role in the securiy of wireless sensor networks, especially for mitigating control and data traffic attacks. Simple neighbor discovery has been found to significantly mitigate the wormhole attack in static sensor networks, [9].

Since neighbor discovery is the first step performed by a sensor node upon deployment and since neighbor discovery requires a very small amount of time, it might be difficult for an adversary to compromise a lot of nodes before neighbor discovery is performed by the entire network. But the compromise of even a single node during neighbor discovery can prove significantly advantageous to the adversary to attack a variety of existing routing protocols. Also, even external malicious nodes (nodes that do not possess the cryptographic keys) can significantly affect neighbor discovery protocols. They just need to relay packets between two non-neighboring nodes and make them believe that they are neighbors. False neighbor discovery will also make protocols that trust on accurate neighbor discovery, like protocols that fight against wormhole attacks, [13], [14], [15], [16] and certain routing protocols completely useless. To understand this, let us consider the following examples.

Let there be two legitimate sensor nodes, A and B, which are not within communication range of each other and an adversary M which is within communication range of both A and B, as shown in Figure 4. During neighbor discovery phase, M can fool A and B to believe that they are neighbors by relaying packets between them. After neighbor discovery, since A and B believe that they are neighbors, all communication between them gets controlled by the adversary M. If M colludes with another malicious node, the situation becomes worse. Colluding malicious nodes can make even legitimate nodes that are very far from each other to believe that they are neighbors. This is illustrated in Figure 5. Once a malicious node or a

set of colluding malicious nodes make two non-neighbor legitimate nodes to believe that they are neighbors, they can easily create a wormhole and launch a variety of attacks against the data traffic flowing on the wormhole, such as selectively dropping the packets.

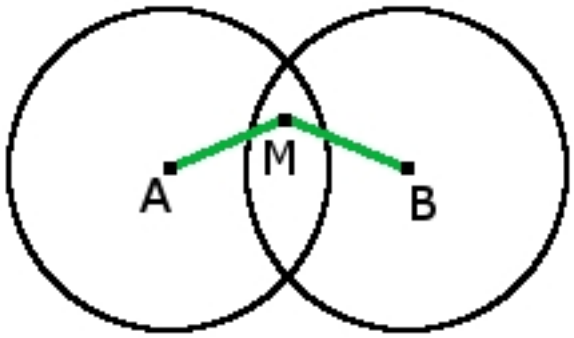


Figure 4: A malicious node M, fooling two legitimate non-neighbor nodes A and B to become neighbors. The communication range of A and B have been abstracted using circles of equal radii.

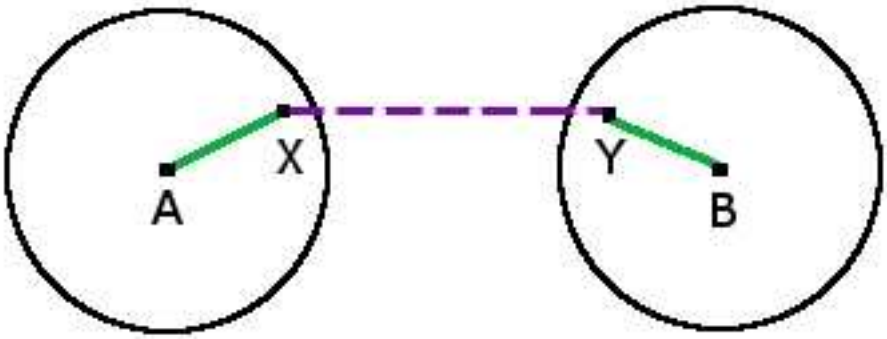


Figure 5: Two malicious nodes, X and Y, fooling two nodes A and B, which are far away, to become neighbors. The communication range of A and B have been abstracted using circles of equal radii.

Therefore, secure neighbor discovery is of immense importance in WSNs. Research on this topic can be broadly classified into three kinds of approaches to this problem. The first approach assumes that there exists no malicious nodes during the neighbor discovery phase due to which neighbor discovery is always secure and using this assumption, it proposes protocols to prevent other attacks ([13], [14], [15], [16]). The second approach performs secure neighbor

discovery in the absence of wormhole attacks (for example - [26], [33]). Such an approach is obviously not secure since even a single external malicious node can prevent neighbor discovery from being accurate. The final kind of approach takes the wormhole attack into consideration while performing neighbor discovery but it either requires specialized hardware in the form of directional antenna arrays or tight synchronization which might not be feasible for sensor networks ([11], [9]). More importantly, the directional antenna approach does not solve the problem completely. We will discuss more about these approaches in section 4. The next section talks about the system model and assumptions.

3.2 System Model and Assumptions

System Model: We assume that the links are bi-directional and the antennas on sensor nodes are omnidirectional. Our protocol does not require the sensor nodes to have any specialized hardware such as GPS or directional antennas. Additionally, the protocol does not require a trusted base station. So, it can be used for neighbor discovery even in applications that function without a base station. However, the protocol does require a pair-wise key management protocol (for example, key pre-distribution techniques as presented in [5], [6], [20]). If the environment is secure enough so that it is not possible to compromise two sensor nodes and make them collude before the neighbor discovery phase gets completed, the key-predistribution protocol proposed in section 2 would be ideal. We assume that all sensor nodes are static and we do not discuss about incremental node deployment at this stage. We also assume that the sensor nodes are randomly distributed in the sensor field. Malicious nodes may be either external nodes (that do not possess the cryptographic keys) or insider nodes (that have been compromised by the adversary). We assume that malicious nodes (both external and internal) do not possess any specialized hardware, such as out-of-band channels or high powered transmission till our protocol completes. Since, as we shall see later, our protocol takes a very short time for neighbor discovery, it is reasonable to assume that an adversary cannot deploy powerful nodes with such specialized hardware before neighbor discovery is finished.

Attack Model: The adversary can eavesdrop on the communication, tamper messages and can relay neighbor discovery information between two non-neighbor nodes and make them

believe that they are neighbors (a form of wormhole attack). The malicious node compromised by the adversary can collude with other malicious nodes and can even make nodes that are far away to believe that they are one-hop neighbors. Essentially, the main intention of a compromised malicious node would be to expand its neighbor list and also the neighbor lists of other nodes and make as many non-neighbor nodes as possible to become neighbors so that it could launch devastating attacks against the network in the future. We also do not protect against Sybil attacks.

3.3 The Neighbor Discovery Protocol

3.3.1 The overhearing technique

The advantage of using omnidirectional antennas is that, when a node sends a packet, all its neighbors can hear the node sending the packet. The identity of the node can be verified using existing cryptographic techniques. Such a technique can be used to verify whether or not a link exists between two nodes. In order for a node to verify whether a link exists between two nodes, it must be within the communication range of both the nodes. We call such nodes as *verifiers*. In order to perform link verification, each node requires two pieces of information.

- Each node needs to find the nodes that claim to be its neighbors.
- Each node needs to know the neighbors of each of its neighbors.

Neighbor verification can then be performed to determine whether the nodes that claimed to be neighbors of a particular node are actually its neighbors.

But for the purpose of monitoring a link, for example, in a protocol like LiteWorp [13], each node also needs to determine the actual neighbors of each of its neighbors. In order for a node, X, to verify whether its neighboring node, Y, is actually transmitting to one of the neighbors of Y (say Z), X also needs to know the neighbor list of Z. Then, X will know the verifiers of the link from Y to Z and can hence use their response to determine whether the link from Y to Z actually exists.

The neighbor discovery protocol is divided into two phases.

1. The Neighbor Discovery Phase

2. The Neighbor Verification Phase

We now describe the Neighbor Discovery Phase.

3.3.2 The neighbor discovery phase

Determination of the expected 1-hop neighbors In this phase, each node finds the nodes that claim to be its neighbors. Upon deployment, each node broadcasts a *Hello* packet and its node ID. Every node that hears this *Hello* packet sends back its ID and a reply containing a nonce which is authenticated using the key that is shared between the nodes. This key is derived using the two node IDs. The initiating node accepts all replies that arrive within a timeout and then authenticates itself to each of its neighbors one by one by sending a hash value of the nonce that they received and adds them to its neighbor list. We call this neighbor list as the *expected neighbor list*. This list might consist of nodes that are not actually within the one-hop communication range of the initiating node. This is because a malicious node which is a neighbor of both the initiator and a non-neighboring node could have fooled both to become neighbors. But, by building this list, every legitimate node within the one-hop communication range of the initiating node gets added in this list. Therefore, the actual list of neighbors is a subset of this *expected neighbor list*.

Determination of the expected 2-hop neighbors Once each node has found its expected list of neighbors, they need to know the neighbors of each of the nodes in this list to determine the *verifiers*. The *verifiers* will be used in the Neighbor Verification phase to decide whether two nodes are actually neighbors. We propose the following simple protocol in order to determine the verifiers.

Each node generates a random key, K, and encrypts its expected neighbor list using K. Each node then does an one-hop authenticated broadcast of its encrypted expected neighbor list. One-hop authenticated broadcast can be easily done using protocols like $\mu - tesla$, [28], or as suggested in [15], [22]. After broadcasting, each node waits to receive the corresponding expected neighbor list of each of its expected neighbors. Once it receives the expected neighbor list of each of its expected neighbors, it does an one-hop authenticated broadcast of the key K. If it does not receive the list within a timeout, it discards the node from its expected

neighbor list, and does an one-hop authenticated broadcast of the key K and the discarded nodes.

This protocol can be easily extended so that a node can also know the verifiers of the link between its neighbor (say X) and the neighbor of X. By doing this, the node can verify whether the nodes that X claims to be its neighbors, are actually the neighbors of X. We now explain how the protocol is extended for this purpose.

After having received the expected neighbor list of each of its expected neighbors, each node, instead of revealing the key K and the dropped neighbors, generate a new key K'. The expected neighbor list of each expected neighbor is encrypted with K'. A single hop authenticated broadcast of this list is sent by each node. After doing this, the nodes, once again, wait to receive the expected neighbor list of each of its expected neighbors. If a node does not send this list within a timeout, it will be dropped from the expected neighbor list of its expected neighbors. After receiving these two lists, each node reveals the keys K, K', the dropped neighbors and the keys revealed by each of its expected neighbors.

The Neighbor Discovery Phase can be summarized as follows:

Table 1: The Neighbor Discovery Phase

Determining the one hop expected neighbors
1. $S \rightarrow$ One hop broadcast: HELLO, ID_S.
2. $X \rightarrow S$: ID_X, $K_{X,S}$(HELLO reply, nonce N).
3. $S \rightarrow X$: $K_{X,S}$(Ack, $h(N)$).
4. S: Adds the ID of X to its expected neighbor list, $NL(S)$.
5. S: Repeats steps 2, 3 and 4 for every HELLO reply received.
Determining the expected two hop neighbors
1. S: Generate key $K_{S,Bcast}$.
2. $S \rightarrow$ One hop broadcast: $K_{S,Bcast}(NL(S))$.
3. S: Wait for $min(T_{out}, NL(T)\ \forall\ T \in NL(S))$.
4. S: Drop nodes that do not send their expected neighbor list within T_{out}.
5. S: Generate key $K'_{S,Bcast}$.
6. $S \rightarrow$ One hop broadcast: $K'_{S,Bcast}(K_{T,Bcast}(NL(T))\ \forall\ T \in NL(S))$.
7. S: Wait for $min(T'_{out}, NL(NL(T))\ \forall\ T \in NL(S))$.
8. S: Drop nodes that do not send their neighbors' neighbor list within T'_{out}.
9. $S \rightarrow$ One hop broadcast: $K_{S,Bcast}$.
10. S: Wait to receive $K_{T,Bcast}\ \forall\ T \in NL(S)$.
11. $S \rightarrow$ One hop broadcast: $K'_{S,Bcast}$, $K_{T,Bcast}\ \forall\ T \in NL(S)$ and dropped neighbors.

At the end of this phase, each node S knows $NL(S)$ and $\forall T \in NL(S)$, S knows $NL(T)$

and $NL(NL(T))$.

3.3.3 The neighbor verification phase

Once each node has completed the neighbor discovery phase, it can determine the verifiers for each of its links. Furthermore, it can also determine the links for which it is a verifier of and who the other verifiers of the link are.

In this phase, we need each node to explicitly announce the destination to which it sends the verification packet. We now describe the neighbor verification phase.

Each node checks whether each of its links has atleast k verifiers. If there doesn't exist atleast k verifiers for a link, the link is dropped. Every verifier of a link also performs this operation. Let N_1 and N_2 be two expected neighboring nodes with atleast k verifiers. N_1 initiates the link verification process by sending an authenticated packet to N_2 and explicitly announcing the address of N_2. Upon receiving the packet, N_2 sends back an authenticated reply to N_1 verifying the link. N_2 also performs a similar operation.

The verifiers that hear the transmission from N_1 hear whether the node that they believe to be N_2 relays the packet to some other node or replies back to N_1. Similarly the verifiers that hear the transmission from N_2 hear whether the node that they believe to be N_1 relays the packet to some other node. Since the neighbor list that was built during the neighbor discovery phase is not necessarily accurate, the verifier list that was built need not be accurate. Therefore, there might exist some verifiers which actually might not hear either the transmission from N_1 or the transmission from N_2 or both. These verifiers mark themselves as *Dropped verifier* for that particular link. If a verifier hears both transmissions and does not detect any packet relaying, then it marks *Link Correct* for that link. If a verifier detects packet relaying, then it marks *Packet Relayed* for that link. Since each node itself is a verifier of the link between itself and its neighbor, if it detects that its packet is being relayed to some other node, it immediately drops the link irrespective of what the other verifiers mark for that link.

The Neighbor Verification phase can therefore be summarized as follows:

We now describe the response algorithm which is finally used by each node to determine its actual neighbors.

Table 2: The Neighbor Verification Phase

1.	S: Determine verifiers, $V_{S\leftrightarrow T}$, $\forall\ T \in NL(S)$.
2.	S: $\forall\ T, U \in NL(S)$, if $T \in NL(U)$ and $U \in NL(T)$, $S \in V_{T\leftrightarrow U}$.
3.	$S \rightarrow T$: $K_{S,T}$(Nonce N) $\forall\ T \in NL(S)$.
4.	$V_{S\leftrightarrow T}$: Hear whether the packet is relayed to T. If yes, mark *Packet Relayed*. If the packet sent by S is not heard, mark *Dropped Verifier*. Else, don't mark anything at this point.
5.	$T \rightarrow S$: $K_{S,T}(h(N))$.
6.	$V_{S\leftrightarrow T}$: Hear whether the packet is relayed to S. If yes, mark *Packet Relayed*. If the packet sent by T is not heard, mark *Dropped Verifier*. Else, don't mark anything at this point.
7.	$V_{S\leftrightarrow T}$: If *Dropped Verifier* has been marked in either of Step 4 or Step 6, mark *Dropped Verifier*. Else, if *Packet Relayed* has been marked in atleast one of Step 4 or Step 6, mark *Packet Relayed*. Else, mark *Link Correct*.

3.4 The Response Algorithm

After the Neighbor Verification phase, each node would have either marked *Dropped Verifier* or *Link Correct* or *Packet Relayed* for every link for which it is a verifier. Each node also knows its expected neighbors as well as the expected neighbors of each of its expected neighbors.

A verifier, V, that has marked *Link Correct* or *Packet Relayed* for a link $A - B$ during the Neighbor Verification phase, first determines whether it has marked *Link Correct* for the links $V - A$ and $V - B$. If it has marked anything else for these two links, it changes its response to *Dropped Verifier*.

After doing this, for each link $A-B$, A, B and the verifiers of the link $A-B$, communicate their response for that link to each of the expected neighbors of A and B. Now each expected neighbor of A and B can determine whether the link $A - B$ exists.

Each node then determines its actual neighbors and the neighbors of its actual neighbors using the following algorithm:

Two nodes will finally be allowed to become neighbors only if all of the following conditions hold:

1. Both nodes claim that their packet was not relayed.

2. After removing verifiers that have marked themselves as *Dropped Verifier*, there still exists atleast k verifiers for that link.

3. Out of the k verifiers, there exists less than γ verifiers that have marked *Packet Relayed* for that link.

3.5 Analysis

3.5.1 Security analysis

We start by defining certain terms that will be useful in proving our results.

Malicious Path: A malicious path between two nodes is a path that consists solely of malicious nodes, except possibly the two end-points.

False Verifier: A false verifier of a link between two nodes claiming to be neighbors, is a node that is present in the expected neighbor list of both the nodes but is not an actual neighbor of atleast one of the nodes.

True Verifier: A true verifier of a link between two nodes claiming to be neighbors, is a node that is an actual neighbor of both the nodes.

Lemma 3.5.1:

The Neighbor Discovery protocol prevents two non-neighboring legitimate nodes from being fooled to become neighbors by the adversary, in the absence of collisions.

Proof:

Let L_1 and L_2 be two non-neighboring legitimate nodes. We consider the following two cases.

Case 1: There exists no malicious path from L_1 to L_2.

In this case, during the Neighbor Discovery phase, L_2 cannot receive the *Hello* packet broadcasted by L_1 and vice-versa. This is because, a legitimate node would never forward the *Hello* packet broadcasted by another node.

Therefore, the expected neighbor list of L_1 will not contain L_2 and vice-versa.

Therefore, L_1 and L_2 won't become neighbors.

Case 2: There exists a malicious path from L_1 to L_2.

In this case, the packets could be relayed between L_1 and L_2 through the malicious path during the Neighbor Discovery phase, due to which L_1 and L_2 could be present in the expected neighbor lists of L_2 and L_1 respectively.

Let the path be $L_1 - M_1 \sim M_2 - L_2$.

Then, during the Neighbor Verification phase, after L_1 sends its verification packet, it will hear M_1 relaying this packet. Similarly, when L_2 sends a reply acknowledging the verification packet sent by L_1, it will hear the reply being relayed by M_2. Obviously, if M_1 and M_2 can communicate using out-of-band channels or directional antennas, the relaying attack will not be overheard by L_1 or L_2. L_1 (or L_2) also might not detect the relaying attack if they experienced a collision when M_1 (or M_2) was relaying the packet.

Similarly, L_2 will hear M_2 relaying its verification packet to L_1 and L_1 will hear M_1 relaying its reply.

Since both L_1 and L_2 are legitimate and they detect the relay, they would mark *Packet Relayed* for the link between them. Therefore, the link would be dropped.

Lemma 3.5.2:

The Neighbor Discovery protocol prevents two non-neighboring nodes, one of which is legitimate and the other malicious, from becoming neighbors, in the absence of collisions.

Proof:

The proof is similar to that of *Lemma 3.1*, except that only the legitimate node will now mark that the link between itself and the malicious node does not exist. The malicious node might or might not mark that the link does not exist.

But since we assume that the links are bi-directional, it is enough for one node to claim that the link does not exist, in order to drop the link.

Thus, the Neighbor Discovery protocol prevents two non-neighboring nodes from becoming neighbors even when one of them is malicious.

Lemma 3.5.3:

The Neighbor Discovery protocol, in the absence of collisions, prevents two malicious non-neighboring nodes, M_1 and M_2, from convincing its legitimate neighbors that they are neighbors under the following conditions:

1. There must be at most $k-1$ compromised nodes that collude with M_1 and M_2 and
2. If there exists v ($> k$) verifiers for the claimed link, t ($< k$) of which are malicious, u of which are legitimate false verifiers and the rest ($v - t - u$) are legitimate true verifiers, then atleast γ out of the $v - t - u$ legitimate true verifiers must have accepted that the link between them and M_1 and M_2 exists and they must also hear atleast one node in each of the malicious paths between M_1 and M_2 relaying packets between M_1 and M_2.

Proof:

Let the two malicious nodes be represented by M_1 and M_2.

We prove that the first condition is necessary, by contradiction. Lets assume that there exists atleast k compromised nodes that collude with M_1 and M_2. Since M_1 and M_2 cannot convince their neighbors that they are actual neighbors, atleast one of M_1 or M_2 should have marked that they are not neighbors or there must have been atleast γ verifiers that have marked *Packet Relayed* for the link $M_1 - M_2$. The former case need not occur since both M_1 and M_2 are malicious, while in the latter situation, M_1 and M_2 will make only malicious nodes to be the verifiers of their claimed link. Since there exists atleast k compromised nodes that collude with M_1 and M_2 in the WSN, by making only malicious nodes as the verifiers, M_1 and M_2 can convince their actual neighbors that the link between them exists, which is a contradiction. Therefore, there must be less than k compromised nodes that collude with M_1 and M_2 in the WSN.

For M_1 and M_2 to become neighbors, there must be atleast k verifiers which might comprise of legitimate true verifiers, malicious true verifiers and malicious false verifiers, out of which, less than γ verifiers should report *Packet Relayed*. Legitimate false verifiers will not become neighbors of atleast one of M_1 or M_2 during the response algorithm and hence won't become verifiers for the link between M_1 and M_2.

Now lets consider the following cases.

Case 1: M_1 and M_2 are more than two hops away from each other.

If M_2 cannot be reached from M_1 in a minimum of two hops, then there is no node that is a neighbor of both M_1 and M_2, because, if there were a node, X, that was a neighbor of both M_1 and M_2, then M_2 can be reached from M_1 in the two hop path, $M_1 - X - M_2$. Therefore, the number of true verifiers for the link $M_1 - M_2$ would be zero. Consequently, the only verifiers for the link $M_1 - M_2$ would be malicious and legitimate false verifiers. Since the legitimate false verifiers will mark themselves as a *Dropped Verifier* due to the response algorithm, the only verifiers, whose response will be considered by the other neighbors of M_1 and M_2 to determine if the link $M_1 - M_2$ exists, are the malicious false verifiers. But since the number of malicious verifiers is less than k, M_1 and M_2 cannot convince their other neighbors that a link exists between them.

Case 2: M_1 and M_2 are two hops away.

This means that there exists atleast one node (legitimate or malicious) that is within the communication range of both M_1 and M_2.

Let there exist a malicious path between M_1 and M_2. Note that this malicious path need not necessarily be a two-hop path. Let there exist $v - t - u$ legitimate true verifiers for the link $M_1 - M_2$. For M_1 and M_2 to not be able to convince their neighbors that the link $M_1 - M_2$ exists, atleast γ out of $v - t - u$ legitimate true verifiers must mark *Packet Relayed* for that link. But for these verifiers to not change their response from *Packet Relayed* to *Dropped Verifier* when they execute the response algorithm, they should accept that the link between them and each of M_1 and M_2 exists. Also, if these legitimate true verifiers are to mark *Packet Relayed* for $M_1 - M_2$, then they should hear the packet sent by M_1 (or M_2) being relayed to M_2 (or M_1) by a malicious node in the malicious path between M_1 and M_2.

If there exists more than one malicious path between M_1 and M_2, then atleast γ legitimate true verifiers must be able to hear atleast one other node in each of these malicious paths between M_1 and M_2. This is because, if there exists a malicious path between M_1 and M_2 in which none of the nodes can be heard by atleast γ legitimate true verifiers, M_1 and M_2 will then use this path to relay packets between them and convince their neighbors that the link $M_1 - M_2$ exists. Consequently, the second condition is also necessary.

This proves the lemma.

Theorem 3.5.1:

The proposed Neighbor Discovery protocol prevents two non-neighboring nodes from convincing their other neighbors that a link exists between them (in the absence of collisions), if atleast one of the following conditions hold:

1. Atleast one of the nodes is legitimate.

2. If both the nodes (say M_1 and M_2) are malicious, there must exist at most $k-1$ compromised nodes that collude with these two malicious nodes and if there exists v ($> k$) verifiers for the claimed link, t ($< k$) of which are malicious, u of which are legitimate false verifiers and the rest ($v-t-u$) are legitimate true verifiers, then atleast γ out of the $v-t-u$ legitimate true verifiers must have accepted that the link between them and M_1 and M_2 exists and they must also hear atleast one node in each of the malicious paths between M_1 and M_2 relaying packets between M_1 and M_2.

Proof:

The proof is a direct consequence of lemmas *3.5.1, 3.5.2* and *3.5.3.*

Corollary:

Each node, upon completion of the Neighbor Discovery protocol, will have knowledge of the following information, in the absence of collisions.

1. The neighbors of a legitimate node, S, will only be those nodes that are within the one-hop communication range of S. The legitimate nodes that are neighbors of a malicious node, X, will only be those nodes that are within the one-hop communication range of X.

2. For a legitimate node, S, let $T \in NL(S)$ and $V \in NL(T)$. If either of T or V are legitimate, S will accept the link $T-V$ to exist, only if V is within the one-hop communication range of T. If both T and V are malicious, S will take the same decision on the link $T-V$ as taken by T, V and the verifiers of the link $T-V$.

Security attacks against the protocol:

We will now describe the security attacks that this protocol is vulnerable to.

1. **Attacks that prevent overhearing:**

   Since our protocol relies on overhearing packet relays, any attack in which a node is not able to hear another node within its communication range is harmful. Two such attacks are described below.

   (a) **Out of band channel attacks:**

   If the adversary replaces the compromised node with a powerful node possessing the capability of transmitting using an out of band channel, then the verifiers will not be able to overhear the packet being relayed by the malicious node. For this attack to be launched, atleast two malicious nodes (internal or external) need to have this capability.

   (b) **Attacks with directional antennas:**

   If the adversary uses nodes that possess directional antennas, only a fraction of the verifiers will be able to overhear the packet being relayed by the malicious node. Therefore, the performance of the protocol will be significantly affected. Even one malicious node is enough to launch this attack. For example, in Figure 6, the malicious node X possesses directional antennas. So, when X relays the packet sent by A to B, only the fraction of nodes present in the shaded portion would be able to overhear this relaying.

2. **Sybil attacks:**

   Let there be two compromised colluding nodes, M_1 and M_2. Let L be a neighbor of M_1 and not a neighbor of M_2. Since M_1 and M_2 collude, they can share their authentication keys between themselves. Then, M_1 can make L believe that M_2 is its neighbor, by claiming the identity of M_2. This is a typical case of a Sybil attack. But, basically, L has only become a neighbor of a node that is within its communication range, but claiming multiple identities. Our neighbor discovery protocol cannot protect against such attacks, primarily because, there is no way of distinguighing between M_1 and M_2.

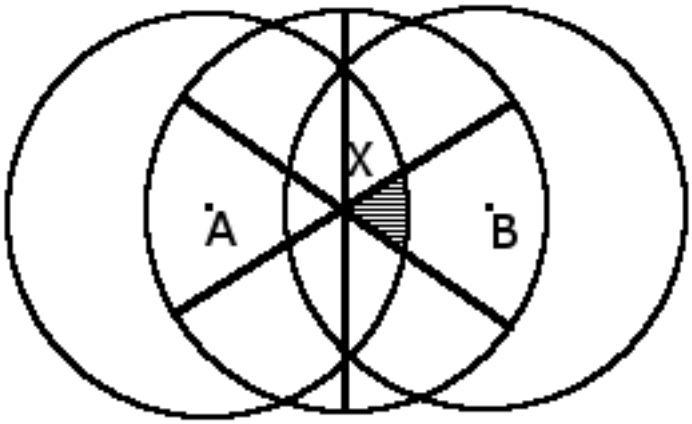


Figure 6: A malicious node X possessing directional antennas

But such attacks can be prevented from damaging the network later on, using protocols that defend against Sybil attacks, for example, [24].

3. **Denial of service attacks:**

   Our protocol does not protect against brute force denial of service attacks like physical layer jamming or physically destroying nodes. It also does not protect against attacks in which the adversary tries to prevent two neighboring nodes from becoming neighbors. We will describe more about this attack in section 4 when we compare our protocol with the directional antenna protocol [9].

3.5.2 Coverage analysis

We now analyze how the protocol performs in the absence of relaying attacks, in terms of the number of legitimate links dropped because of the non-existence of k verifiers, so that we can empirically find a good value of k that will provide good coverage. We still assume that malicious nodes exist in the WSN, but that they only indulge in passive attacks.

We abstract the communication range of each sensor node in the WSN by a circle of radius, r. Let us have two neighboring nodes, A and B, separated by a distance d. Then, the verifiers of the link between A and B are those nodes that are present in the shaded region in Figure 7. The area of this shaded region is given by

$$Ar_{omni} = 2r^2 \cos^{-1}(\frac{d}{2r}) - \frac{d}{2}\sqrt{4r^2 - d^2} \tag{6}$$

Suppose there are a total of N sensor nodes uniformly and randomly distributed in a sensor field of area, Ar. For the link between A and B to exist, we need atleast k verifiers in the shaded region in Figure 7.

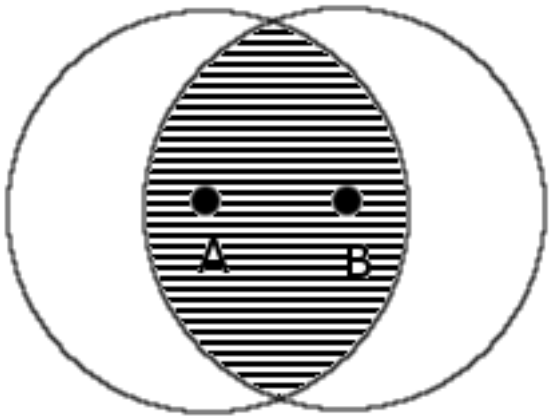


Figure 7: Atleast k nodes need to be neighbors of both A and B

The probability that atleast k verifiers are present in the shaded area in Figure 7 is given by

$$P(\{Atleast\ \ k\ \ verifiers\ \ in\ \ Ar_{omni}\}) = \sum_{i=k}^{N} \binom{N}{i} (\ \tfrac{Ar_{omni}}{Ar}\ )^i (\ 1 - \tfrac{Ar_{omni}}{Ar}\ )^{N-i} \tag{7}$$

Ar_{omni} is minimum when B is on the edge of the communication range of A, so that, $d = r$. This area is given by

$$Ar_{omni} = r^2\ (\ \tfrac{2\pi}{3} - \tfrac{\sqrt{3}}{2}\ ) \tag{8}$$

Assuming 100 nodes in the WSN with an average of 10 neighbors for each node and with a communication range of 30 m, Figure 8 shows the probability of having atleast k verifiers in this minimum area, for varying k.

Thus, with probability > 0.8, there exists atleast 3 verifiers for a link and with probability

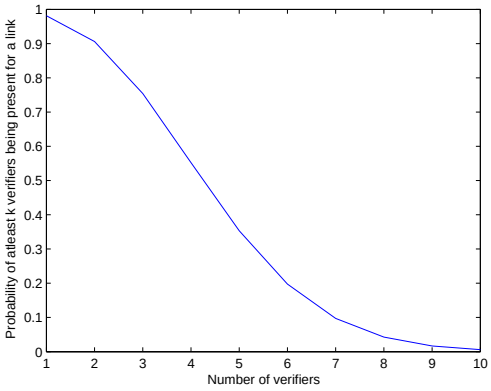


Figure 8: Probability that there exist atleast k verifiers for a link

> 0.9, there exists atleast 2 verifiers for a link. Since the areas that we have taken into consideration occur when the two nodes are at the edge of each other's communication range, this probability is actually a lower bound.

We have analyzed the protocol and provided results in the absence of collisions. The analysis, in the presence of collisions, will be similar to that provided in [13] and will, therefore, not be discussed here.

3.6 Simulations

The simulation is performed using MATLAB [23]. Sensor nodes are uniformly and randomly deployed in a 100×100 square field. The number of nodes in the field vary from 10 to 100. Table 3.3 lists the simulation parameters.

Table 3: Simulation Parameters

Communication range	$30m$
Number of nodes	$10 - 100$
Sensor field size	$100 \times 100m^2$

Since our protocol requires each legitimate link to have k verifiers for the link to exist, the fraction of legitimate links that get dropped due to the non-existence of k verifiers for certain links, is simulated for different k. This simulation is done in the absence of malicious nodes. The simulation is run a 1000 times and the results are averaged. Since, in our protocol, each node, itself, is a verifier of the link that it is a part of, no links get dropped when $k = 1$. This can be seen in Figure 9. For $k > 1$, the fraction of links dropped decreases as the density of

the sensor nodes in the WSN increases, which is obvious, since the probability that k verifiers exist for a link increases as the node density increases.

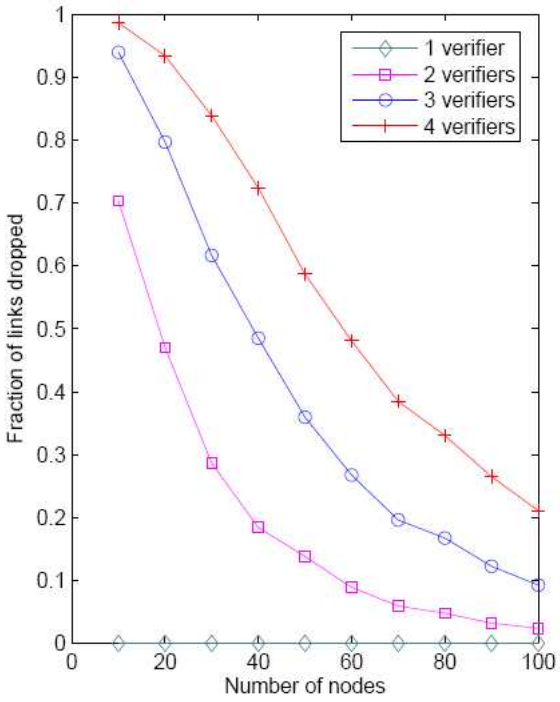


Figure 9: Fraction of links dropped due to the necessity of the existence of k verifiers for every link

Next, the amount of storage in each sensor node after the end of the neighbor discovery phase is simulated. This storage includes the neighbors of the node, the neighbors of the first hop neighbors and the neighbors of the second hop neighbors of the node. Here, the number of malicious nodes in the network is varied. This simulation is performed for 40 nodes (corresponding to an average of approximately 11 neighbors a node) distributed in the 100×100 sensor field. It is assumed that the malicious nodes relay the *Hello* packets that they hear, to all their neighbors. The simulation is again run a 1000 times and the results are averaged. Figure 10 shows the storage (in KB) in each of the 40 nodes, assuming that each node ID requires 4 bytes.

Figure 11 shows the size of the expected neighbor list at each node when the number of malicious nodes in the network varies.

We see that the average number of neighbors present in the expected neighbor list of a

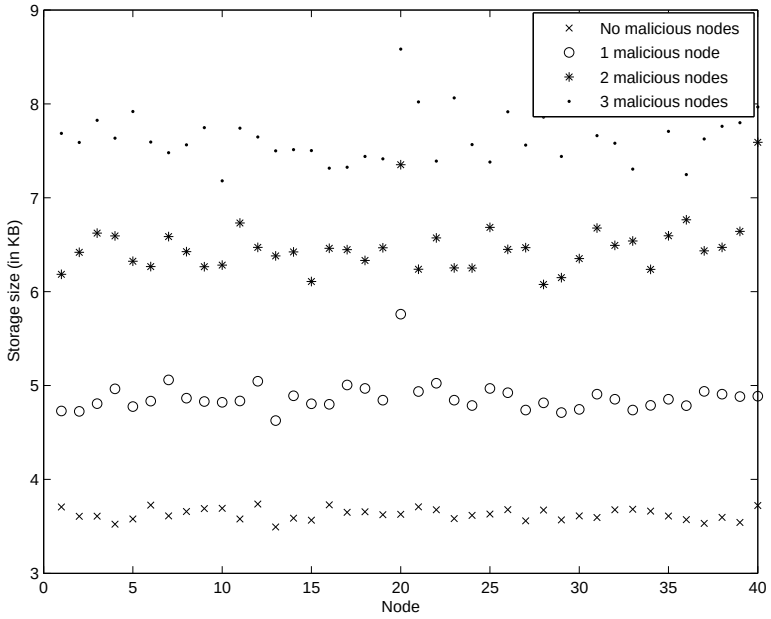


Figure 10: Total storage (in KB) of neighbor lists at each node

node increases very little as the number of malicious nodes in the WSN increases. On the average, for a network with 40 nodes in a 100×100 field (corresponding to an average of 11 neighbors a node), the maximum number of neighbors that any node possesses is around 14, in the absence of malicious node. For one, two and three malicious nodes, the maximum number of nodes in the expected neighbor list increases to 16, 18 and 19 respectively.

Therefore, the protocol does not require much storage and is suitable for a WSN.

4 Extensions and Comparisons

A secure neighbor discovery protocol for static sensor networks in which all the sensor nodes are deployed initially, has been proposed and analyzed in the previous section. In a static sensor network, incrementally deploying nodes provides a lot of flexibility in deployment. It is also advantageous in the sense that, malfunctioning nodes or nodes that have died out could be removed and new nodes could be deployed in their place. The first part of this section therefore suggests possible extensions to the neighbor discovery protocol so that it would handle incremental deployment of nodes as well.

The second part of this section suggests modifications to the protocol in order for it to

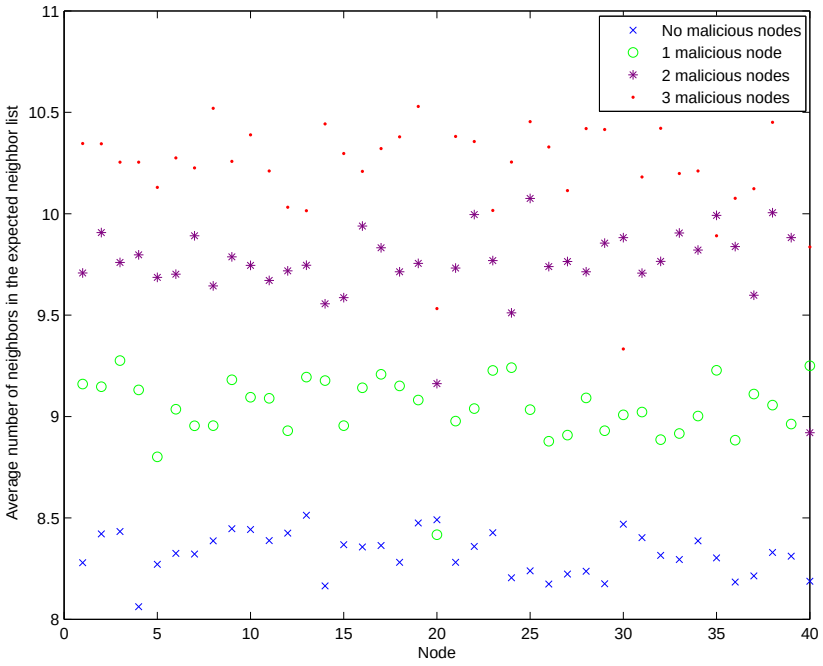


Figure 11: Average number of neighbors in the expected neighbor lists of nodes in the presence of malicious nodes

work when the sensor nodes are mobile. Since sensor nodes are deployed randomly, mobile sensor nodes could move around, connecting disconnected regions and thus improving the coverage. Therefore, mobility is an important constituent in WSNs.

Finally, this section compares our protocol with three related protocols and analyzes the advantages and disadvantages of using these protocols.

4.1 Incremental Deployment

4.1.1 Assumptions

Our attack model is the same as that described in section 3. For the system model, we make the additional assumption that any node in the WSN can distinguish between an incrementally deployed node and a node that was already present before incremental deployment. ID based authentication protocols, for instance, [8], [4] can easily achieve this. The idea suggested in section 2 could also be used.

4.1.2 The protocol

With the additional assumption that we have made, the same neighbor discovery and neighbor verification protocol could be directly used for incrementally deployed nodes as well.

When nodes are incrementally deployed, some nodes in the neighborhood of the newly deployed nodes would have been deployed much before and would have already built their neighbor lists while others would have been deployed along with the newly deployed nodes. Those nodes that have already built their neighbor lists only need to send these lists and verify that the link between them and the newly deployed nodes exist. Newly deployed nodes would build expected neighbor lists and would verify each and every link in order to build their first and second hop neighbor lists. To summarize, the protocol consists of the following steps.

1. A newly deployed node will perform neighbor discovery and neighbor verification as described in section 3.

2. An already existing node that is present in the expected neighbor list of the newly deployed node will broadcast the expected neighbor list of the newly deployed node to its neighbors and will send the neighbor lists of each of its neighbors to the newly deployed node.

3. An already existing node that is two hops away from the newly deployed node will send its neighbor list to the newly deployed node.

4.2 Mobility

4.2.1 Suggestions

The need to differentiate between a newly deployed node and an already existing node becomes extremely important in the presence of mobility. For example, in the MobiWorp protocol [14], a node, in order to move from one location to another, needs a certificate from a central authority that it can use to integrate itself in the new location. But a malicious node that had been locally isolated, could move from one location to another and claim in the new location that it is a new node that has been deployed in the network. ID based authentication protocols

would prevent such claims from fooling legitimate nodes in the WSN. As a simple example, we will assume that all nodes in the network know the ID of the last deployed node and that the nodes are deployed in a known order of IDs. Then, a node would be able to distinguish between an already existing node and a newly deployed node using ID based authentication.

Protocols, as suggested in [14], can then be used for secure movement of nodes from one location to another. Once nodes have securely moved from one location to another, they can perform neighbor discovery in the same way as incremental nodes perform neighbor discovery. Though the neighbor discovery procedure is the same for both incremental nodes and nodes that have moved from one location to another, the sensor nodes in the network can differentiate between the newly deployed nodes and the nodes that have moved from another location. A trusted central authority would also be required in order to facilitate secure movement of nodes [14] and to differentiate between newly deployed nodes and nodes that already exist.

4.3 Comparison Between Protocols

4.3.1 The directional antenna protocol

Hu and Evans [9] proposed a protocol that uses directional antennas in order to perform secure neighbor discovery in the presence of wormhole attacks. Their protocol is briefly described here.

A sensor node possesses an antenna with N zones. Each zone has a conical radiation pattern, spanning an angle of $\frac{2\pi}{N}$ radians. The zones are fixed with non-overlapping beam directions, so that the N zones may collectively cover the entire plane as shown in Figure 12. The basic idea of the protocol is that when a node A sends a packet directly to a neighboring node B, if node B receives the packet in zone β, then node A should receive the reply sent by B in the radially opposite zone, denoted by $\hat{\beta}$. From now on, zone(A, B) will denote the zone in which A hears B.

Since a malicious node could still fool two nodes that are not actually neighbors to become neighbors if they are in opposite zones (Figure 13(a)), a strict neighbor discovery protocol is proposed in order to overcome this problem (Figure 13(b)). This protocol requires atleast one valid verifier to exist for each link. A verifier of a link $A \leftrightarrow B$ is essentially a node that both

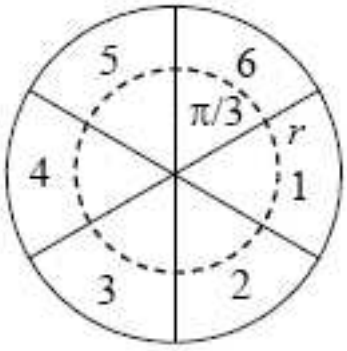


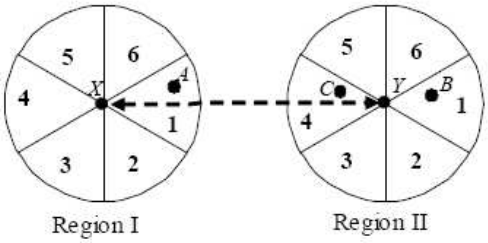
Figure 12: A directional antenna with six zones and with a transmission range of r. This figure has been taken from [9].

A and B believe to be their neighbor. For a node V to be a valid verifier for the link $A \leftrightarrow B$, V must satisfy the following conditions:

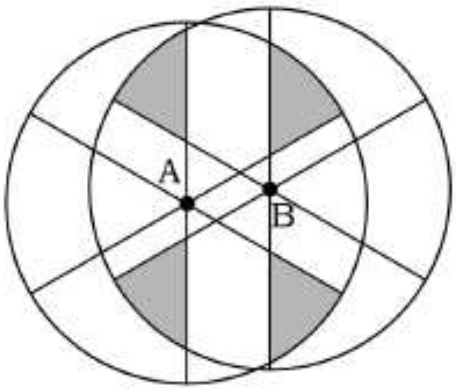
1. zone(B,A) $\neq$ zone(B,V).

2. zone(B,A) $\neq$ zone(V,A).

3. zone(B,V) cannot be both adjacent to zone(B,A) and adjacent to zone(V,A).



(a) Without verifiers, X and Y can still fool A and C to become neighbors.



(b) The strict neighbor discovery protocol.

Figure 13: The functioning of the directional antenna protocol with and without verifiers [9].

From Figure 13(b), it is clear that a valid legitimate verifer cannot exist within the communication range of both B and A if the three conditions hold.

It is claimed that the strict neighbor discovery protocol prevents wormhole attacks when each wormhole has atmost two endpoints. But the following counterexample shows that this

claim is actually not true. The adversary can place two malicious nodes in such a way that two legitimate nodes that are not actually neighbors are fooled to become neighbors.

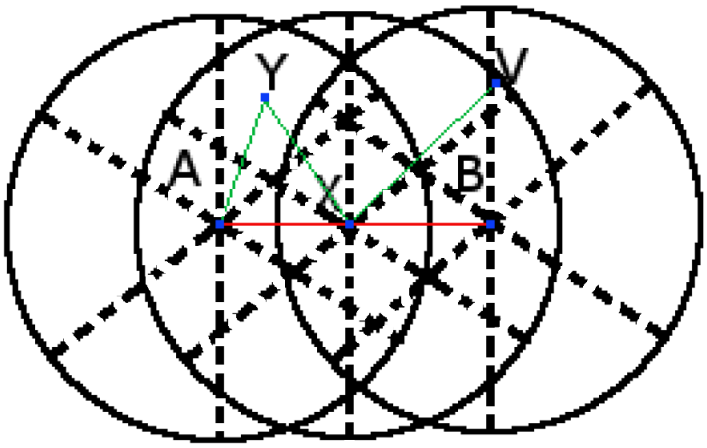


Figure 14: The problem with the directional antenna protocol

In Figure 14, A and B are two legitimate nodes that are not within the communication range of each other. Let the zones be numbered as shown in Figure 12. X and Y are two colluding malicious nodes. V is a legitimate node which will be used by X and Y to fool A and B that a valid verifier exists for the link $A \leftrightarrow B$. Packets from A will be relayed to B through X and since A and B hear each other in opposite zones, they will now look for a verifier to confirm that they are neighbors. Now, V is a node such that zone(B,V) = 3 is opposite to zone(V,A) = 6. Also zone(B,A) = 1. Thus, V satisfies the conditions for a valid verifier. But for V to convince A, they should hear each other in opposite directions. The malicious node Y facilitates this. The transmission from A to V takes the path $A - Y - X - V$. Thus, the two malicious nodes fool A and B to believe that they are neighbors.

Also, the directional antenna protocol only tries to prevent two legitimate non-neighboring nodes from becoming neighbors. Malicious nodes that are far away could easily become neighbors with both legitimate as well as malicious non-neighbors. The directional antenna protocol also does not consider framing attacks in the sense that the verifier itself could be malicious.

Moreover, the necessity for the existence of a verifier in such a small region (Figure 13(b)) in order for a legitimate link to exist, nullifies the advantages of having an increased communi-

cation range. In fact, the directional antenna protocol, with the directional antenna having a communication range that is 1.8 times larger than the omni-directional communication range, drops more links than our protocl that uses omni-directional antennas. This can be seen from Figure 9 and from the results presented in [9].

For a typical neighborhood density of 10 neighbors a node with an omni-directional antenna (corresponding to approximately 33 nodes with a directional antenna), the strict neighbor discovery protocol, with one verifier, drops 40% of the legitimate links [9] while our protocol only drops 25% of the legitimate links, with two verifiers. Since, in our protocol, each node itself is a verifier of the link that it is a part of, our protocol does not drop any links when we need use only one verifier. Thus, our protocol effectively outperforms the directional antenna protocol in this regard.

The directional antenna protocol can however prevent out of band channel attacks to a certain extent while our protocol has absolutely no resistance to these attacks. Our protocol can however prevent wormhole attacks with multiple endpoints while the directional antenna protocol can only prevent wormhole attacks with one malicious node. Table 4.1 summarizes the comparison between these two protocols.

Table 4: Comparison between our protocol and the directional antenna protocol

Property	Our Protocol	The Directional Antenna Protocol
Special hardware	None	Directional antennas
Out of band channel attacks	Vulnerable	Slight resistance
Directional antenna attacks	Vulnerable	Slight resistance
Other wormhole attacks	Resistant to wormholes with multiple endpoints	Resistant to wormholes with only one endpoint
DoS attacks	Vulnerable	Vulnerable
Fraction of links dropped	25% with two verifiers	40% with one verifier
Framing attacks	Resistant	Vulnerable
Miscellaneous	Prevents malicious non-neighboring nodes from becoming neighbors to other nodes	Does not prevent malicious non-neighboring nodes from becoming neighbors to other nodes

4.3.2 A timing based protocol

Hu, Perrig and Johnson [11] had proposed a secure Neighbor Detection protocol that allows both the initiator and the responder to check that the other is within a maximum communication range. A node sends a Neighbor Solicitation packet to a neighbor and gets back a Neighbor Reply packet. The measured delay between sending the first message and receiving the second message is used by the protocol to provide an upper bound on the distance of the neighbor.

Obviously, such an approach, requires very tight timing and since messages travel at the speed of light and due to congestions and collisions in the network, this method would be highly inaccurate and practically infeasible.

4.3.3 Sensor finger-printing

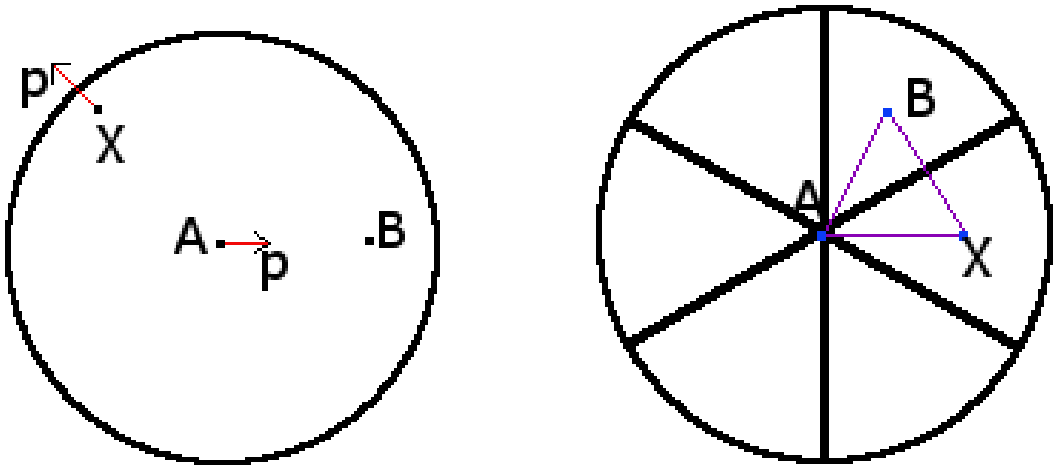
One of the fundamental problems that sensor nodes face is that they cannot differentiate between a node that has relayed a packet and the node from which the packet has originated from. Rasmussen and Capkun, in [30], present a technique called sensor fingerprinting in which each sensor node can identify the node from which it has received a packet using the nature of the signal that it has received. For example, if node B transmits a signal of type b, node A transmits a signal of type a and node X transmits a signal of type x, then node B can differentiate between a signal that it receives directly from A and a signal that has been relayed to it by X. A key idea is that nodes cannot control the type of signal that they transmit. While this may be true for legitimate nodes, its not correct to make such an assumption for malicious nodes as well.

This technique requires advanced physical layer features in each sensor node and is along the lines of using tamper-proof hardware which is very expensive. It also remains to be seen whether malicious nodes can copy the signal sent by nodes, in which case, this technique would be as bad as the others.

4.4 Denial of Service Attacks

In the process of solving the neighbor discovery problem, only one part of the problem has been given adequate importance, i.e., preventing two non-neighboring nodes from becoming neighbors. The problem of ensuring that two neighboring nodes become neighbors is actually a challenging problem in itself. The adversary could launch denial of service attacks against the neighbor discovery protocol to prevent two neighboring nodes from becoming neighbors. Such an attack which would affect our neighbor discovery protocol, is presented in this section.

Lets have two neighboring nodes A and B as shown in Figure 15(a). Let X be a malicious node that is a neighbor of A but not a neighbor of B. When A sends the verification packet, p, to B during the Neighbor Verification Phase, X can generally replay this packet since it can hear A. Since our protocol relies on overhearing techniques, A would think that its packet is being relayed to B and would therefore drop the legitimate link $A \leftrightarrow B$. But it is important to note that this attack would be successful only if X replays the packet before B sends back its reply. Therefore, strict timing requirements are essential for this attack to succeed.



(a) The denial of service attack in our protocol; X replays the verification packet p to confuse A.

(b) The denial of service attack in the directional antenna protocol

Figure 15: Denial of service attacks in secure neighbor discovery protocols.

This problem also exists in the directional antenna protocol. A malicious node X which is in zone 1 of a legitimate node A could forward the *Hello* packet sent by A in zone 1 to a legitimate node B present in zone 6 of A (Figure 15(b)). But since B hears from X in zone

2 and A hears from B in zone 1 through X, X effectively prevents A and B from becoming neighbors.

Protocols that have been proposed to prevent two non-neighboring nodes from becoming neighbors have not considered these kind of attacks in the past and therefore this is worth mentioning.

5 Related Work

5.1 Key Pre-Distribution

It has been generally agreed in the literature that asymmetric key cryptosystems are not suitable for sensor networks ([25], [31], among others) because of the requirement of high computational power. Symmetric key cryptosystems have, therefore, been vastly studied for key distribution and management in sensor networks. Examples of these protocols are the group-based pre-deployed keying, probabilistic key distribution protocols and key distribution protocols that involve significant radio communication.

Eschenauer and Gligor [7] present a probabilistic key pre-deployment scheme for sensor networks. From a large key pool, they select m keys at random and these keys are loaded into each sensor node before deployment. If a common key does not exist between two nodes, then intermediate nodes are used for secure key exchange. Compromising a single node reveals m keys to the adversary. Also, a malicious intermediate node could eavesdrop on all the communication between two nodes that had used it for exchanging keys between them. Chan *et al.* [5] provide an extension to this scheme but their scheme requires significant radio communication.

Blom's scheme [2] has been described in detail in section 2. Grid-based key distribution was first presented in [8]. The sensors are distributed in a logical grid with each sensor having an identity (i, j) where i represents the row and j represents the column in the grid. Keys are shared between every pair of rows and every pair of columns. If a node with identity (i, j) wants to establish a key with node (p, q), then the keys shared between row i and row p and the keys shared between column j and column q are used to derive a unique key between the two nodes. The problem with this approach is that this key will also be known by nodes (i, q)

and (p, j) since they also know the corresponding row and column keys. Also, compromising two nodes would expose a huge fraction of the communication. Chan and Perrig [4] present a different approach of key establishment using a logical grid. Each node, in their scheme, shares a pair-wise key with every other node in its row and in its column. If a node wants to communicate with a node that is present in some other row or column, it has to use an intermediate node in order to securely exchange keys. This involves significant communication overhead and is more secure than the original matrix based scheme [8]. Liu *et al.* [21] modify these approaches by using location-based grids instead of logical grids.

Other approaches in key distribution and management include [33], [28], [35], [16].

While we have concentrated on secure one-to-one communication, broadcast authentication is a different challenging problem. There has been an extensive study on securing broadcast in sensor networks and for further reading, the reader can refer to [27] and [19].

5.2 Neighbor Discovery

As we have seen in section 3, secure neighbor discovery is a critical issue in sensor networks. Neighbor discovery protocols are extremely vulnerable to the wormhole attack. The reader can refer to [12], [29], [10], [11], [13], [14], [30] for research on wormhole attacks and their countermeasures.

Few protocols consider the issue of securing neighbor discovery in the presence of wormhole attacks. The protocol by Hu and Evans [9] uses directional antennas for dynamic neighbor discovery and verification. The scheme by Perrig [11] relies on time of flight and assumes very accurate time measurement in order to verify whether a node is within the communication range of another node. A new approach called sensor fingerprinting [30] has been recently proposed for addressing this issue but it requires advanced physical layer features. A detailed comparison of these protocols with our protocol has been provided in section 4.

The protocols suggested in [13], [14], [15], [16], [18], [1], [32], [36], among others, assume that malicious nodes are not present in the network during the neighbor discovery stage. While it might be difficult for an adversary to compromise a node before neighbor discovery gets completed, even an external malicious node could easily affect the neighbor discovery protocol. Also, this assumption prevents us from incrementally deploying nodes. The scheme

by Lee and Choi [17] assumes that one-hop neighbor discovery is secure and presents an approach, using overhearing, for securely discovering the second hop neighbors.

The schemes proposed in [26] and [33] do not take the wormhole attack into consideration while performing neighbor discovery. A location-based technique has also been proposed for securing neighbor discovery in WSNs [34]. But the location estimation protocol is itself vulnerable to security attacks. Moreover, location-based techniques abstract the communication range of nodes as a circle of radius r. The actual communication pattern of an omni-directional antenna is generally skewed. Therefore, protocols that rely on such an abstraction would not be practically feasible.

6 Conclusions and Future Work

6.1 Conclusions

We have discussed in detail about the neighbor discovery problem from the point of view of security in WSNs. It has explained the importance of secure neighbor discovery and has proposed a secure neighbor discovery protocol. It has also analyzed a whole lot of issues that the current neighbor discovery protocols face.

A simple key pre-distribution has been proposed in section 2. This technique is scalable, memory efficient, distributed and is ideal for sensor networks in the absence of colluding malicious nodes.

The secure neighbor discovery and verification protocol for static sensor networks has been proposed in section 3. The protocol has been analyzed in detail and has been proven, by simulations, to outperform current neighbor discovery schemes.

We have briefly studied about the extension of this protocol to incremental deployment and have also provided suggestions to modify this protocol for mobile sensor networks, in section 4. Further, we have compared between various neighbor discovery approaches in detail and analyzed a whole lot of open issues that the neighbor discovery problem poses.

6.2 Directions for Future Research

There still remains a lot of open issues in the secure neighbor discovery problem. Denial of service attacks and attacks using nodes with powerful hardware capabilities significantly affect the neighbor discovery protocol and are issues to ponder about. Another interesting problem to look at is whether a neighbor discovery protocol could be made secure against Sybil attacks. A malicious node could simply create copies of itself in different parts of the network and become neighbors to a large number of nodes and then launch wormhole attacks.

Yet another issue to be addressed is designing a secure neighbor discovery protocol in mobile sensor networks. A typical approach would be to break this problem into two parts. The first part would solve the problem of secure node relocation and the second part would solve the neighbor discovery problem. If the former problem is solved, our protocol could be easily extended to solve the neighbor discovery problem.

Current secure node relocation approaches involve the presence of a central trusted authority which might not be feasible in all sensor networks. A distributed and secure node relocation approach would solve the neighbor discovery problem for mobile sensor networks and would therefore be a good direction for future research.

References

[1] R. Anderson, H. Chan, and A. Perrig. Key infection: smart trust for smart dust. In *IEEE International Conference on Network Protocols (ICNP 2004)*, 2004.

[2] R. Blom. An optimal class of symmetric key generation systems. In *Proceedings of the EUROCRYPT 84 workshop on Advances in cryptology: theory and application of cryptographic techniques*, pages 335–338, 1985.

[3] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In *Advances in Cryptology (CRYPTO 92)*, pages 471–486, 1993.

[4] Haowen Chan and A. Perrig. Pike: Peer intermediaries for key establishment in sensor networks. In *Proceedings of the* 24^{th} *Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, volume 1, pages 524–535, 2005.

[5] Haowen Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *Proceedings of the Symposium on Security and Privacy*, pages 197–213, 2003.

[6] W. Du, J. Deng, Y. Han, and P. Varshney. A pair-wise key pre-distribution scheme for wireless sensor networks. In *Proceedings of the* 10^{th} *ACM conference on Computer and communication security (CCS'03)*, 2003.

[7] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the* 9^{th} *ACM conference on Computer and communications security*, pages 41–47, 2002.

[8] L. Gong and D. J. Wheeler. A matrix key-distribution scheme. *Journal of Cryptology*, 2:51–59, 1990.

[9] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. In *Network and Distributed System Security Symposium*, 2004.

[10] Y. C. Hu, A. Perrig, and D. B. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *Proceedings of the* 22^{nd} *INFOCOM*, pages 1976–1986, 2003.

[11] Y. C. Hu, A. Perrig, and D. B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *ACM WiSe Workshop*, 2003.

[12] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's AdHoc Networks Journal*, 1:293–315, September 2003.

[13] I. Khalil, S. Bagchi, and N. B. Shroff. Liteworp: A lightweight countermeasure for the wormhole attack in multihop wireless networks. In *Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 612–621, 2005.

[14] I. Khalil, S. Bagchi, and N. B. Shroff. Mobiworp: Mitigation of the wormhole attack in multihop wireless networks. In *Proceedings of the International Conference on Security and Privacy in Communication Networks*, 2006.

[15] Issa Khalil, Saurabh Bagchi, and Cristina Nina-Rotaru. Dicas: Detection, diagnosis and isolation of control attacks in sensor networks. In *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communication Networks (SECURECOMM'05)*, pages 89–100, 2005.

[16] Issa Khalil, Saurabh Bagchi, and Ness B. Shroff. Analysis and evaluation of secos, a protocol for energy efficient and secure communication in sensor networks. *Elsevier Ad-hoc Networks Journal*, 5:360–391, april 2007.

[17] Suk-Bok Lee and Yoon-Hwa Choi. A resilient packet-forwarding scheme against maliciously packet dropping nodes in sensor networks. In *ACM CCS Workshop on Security of Ad Hoc and Sensor Networks (SASN'06)*, October 2006.

[18] Suk-Bok Lee and Yoon-Hwa Choi. A secure alternate path routing in sensor networks. *Computer Communications*, 30:153–165, December 2006.

[19] D. Liu and P. Ning. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networsk. In *Proceedings of the* 10^{th} *Annual Network and Distributed System Security Symposium*, pages 263–276, February 2003.

[20] D. Liu and P. Ning. Establishing pair-wise keys in distributed sensor networks. In *Proceedings of the* 10^{th} *ACM conference on Computer and communication security (CCS'03)*, 2003.

[21] D. Liu, P. Ning, and W. Du. Group-based key pre-distribution in wireless sensor networks. In *Proceedings of 2005 ACM Workshop on Wireless Security (WiSe 2005)*, pages 11–20, September 2005.

[22] D. Liu, P. Ning, S. Zhu, and S. Jajodia. Practical broadcast authentication in sensor networks. In *The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous)*, pages 118–129, July 2005.

[23] MATLAB. The MathWorks Inc.

[24] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks: analysis and defenses. In *Proceedings of the third international symposium on Information Processing in sensor networks*, pages 259–268, 2004.

[25] C. Park, K. Kurosawa, T. Okamoto, and S. Tsujii. On key distribution and authentication in mobile radio networks. *Advances in Cryptology - EuroCrypt '93*, 765:461–465, 1993.

[26] B. Parno, M. Luk, E. Gaustad, and A. Perrig. Secure sensor network routing: A clean-slate approach. In *Proceedings of Conference on Future Networking Technologies (CoNEXT)*, December 2006.

[27] A. Perrig, R. Canetti, J. Tygar, and D. Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, 2000.

[28] A. Perrig, R. Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. Spins: security protocols for sensor networks. *Wireless Networks*, 8:521–534, 2002.

[29] Radha Poovendran and Loukas Lazos. A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks. *Wireless Networks*, 13:27–59, January 2007.

[30] Kasper Bonne Rasmussen and Srdjan Capkun. Implications of radio fingerprinting on the security of sensor networks. In *Proceedings of IEEE SecureComm*, 2007.

[31] M. Tatebayashi, N. Matsuzaki, and D. B. J. Newman. Key distribution protocol for digital mobile communication systems. *Advances in Cryptology - CRYPTO '89*, 435:324–334, 1989.

[32] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbough. Toward resilient security in wireless sensor networks. In *ACM MobiHoc'05*, pages 34–35, 2005.

[33] Y. Zhang, W. Liu, W. Lou, and Y. Fang. Securing sensor networks with location-based keys. In *Wireless Communications and Networking Conference*, volume 4, pages 1909–1914, March 2005.

[34] 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*, 24:247–260, February 2006.

[35] Li Zhou, Jinfeng Ni, and C. V. Ravishankar. Short paper: Gke: Efficient group-based key establishment for large sensor networks. In *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communication Networks (SECURECOMM'05)*, pages 397–399, 2005.

[36] S. Zhu, S. Setia, and S. Jajodia. Leap: efficient security mechanisms for large-scale distributed sensor networks. In *ACM Conference on Computer and Communications Security (CCS'03)*, October 2003.