Secure Network Discovery in Wireless Sensor Networks Using Combinatorial Key Pre-Distribution

Kevin Henry
Cryptography, Security, and Privacy (CrySP) Lab
Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
Email: k2henry@uwaterloo.ca

Douglas R. Stinson
Cryptography, Security, and Privacy (CrySP) Lab
Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
Email: dstinson@uwaterloo.ca

***Abstract*—Many sensor network protocols utilize the existence of disjoint paths (e.g., perfectly secure message transmission or multi-path key establishment), but do not address how a node actually determines these paths in the presence of an adversary. In this paper we investigate what assumptions are necessary to gather information about the local network topology when adversarial nodes are present and capable of lying about their identity or neighbors in the network. These assumptions are practical, and realizable through existing tools such as combinatorial key pre-distribution, fingerprinting, and localization. Our protocols ensure that, except with small probability, if node accepts a path through the network as valid, then each node along that path must be telling the truth about its identity and nodes it can communicate with, so long as a majority of honest nodes are present in the network at each point decisions are made.**

I. Introduction

A sensor network is an ad-hoc network comprised of many computationally limited sensor nodes. In general, these nodes communicate wirelessly, have very limited memory, and possess no external power source. Because of the limitations of sensor nodes, public-key cryptography is not often utilized and protocols are optimized to reduce the amount of communication required between nodes. Martin and Paterson [1] have provided a framework which summarizes the various types of sensor networks.

Sensor networks can be broadly categorized according to their relative capabilities, how they are deployed, and their ideal communication structure. In a *homogeneous* sensor network all nodes are identical, whereas a *hierarchical* network contains nodes of varying capabilities. In practice, this is often in the form of a two or three level tree, with many low power nodes forwarding information to a single higher power node (often called a base station). A large network may have several base stations, all forwarding their data to some final central location.

In terms of deployment, sensor nodes can either be *fixed* or *mobile*. In the latter case, a distinction can be made between nodes that are free to travel throughout the entire network (*fully mobile*), and those nodes may only move within in fixed region (*locally mobile*). When considering fixed networks, distinctions are made between having *full control*, *partial control*, or *no control* over the location of nodes after deployment.

The goal of this paper is to start with a simple homogeneous, fixed, no control network with pre-distributed cryptographic keys and investigate what assumptions are necessary to allow a node to accurately determine the local network topology in the presence of an active adversary. This allows a sensor node to establish multiple disjoint paths between itself and a destination, thus allowing the use of the perfectly secure message transmission protocol of Dolev et al. [2] and other protocols built upon the same idea, such as the multi-path key establishment protocols of Wu and Stinson [3] in order to establish a secure shared key between a node and the desired destination. Determining disjoint paths between two nodes is facilitated through the use of a combinatorial key pre-distribution scheme that allows a node to prove its identity based on which keys it possesses.

We present a protocol that proceeds in multiple steps, each allowing a node to expand its view of the network by one hop. This protocol is based on voting and the assumption that an honest majority of nodes is present at each point in the network a decision must be made about the identity of a node. This is accomplished by the use of a combinatorial key pre-distribution scheme that allows a node to prove its identity by proving that it possesses just two keys from its keylist. In order to evaluate the practicality of our assumptions, we provide simulated results demonstrating that with proper parameter choice we can still determine network topology even when a large number of malicious nodes are present during network discovery.

A. Related Work

In addition to providing a framework for categorizing sensor networks, Martin and Paterson [1] provide a comprehensive survey of key establishment protocols for sensor networks, while Karlof and Wagner [4] have surveyed a number of attacks against routing protocols in sensor networks.

Many secure routing protocols for ad-hoc wireless networks have been presented, such as $ARAN$ [5], S-$AODV$[6], and DV-SRP [7], but have not focused on solving the problem of establishing several disjoint paths to a destination.

Poturalski, Papadimitratos, and Hubaux [8] have given a formal treatment of neighborhood discovery in wireless networks and give an impossibility result for protocols that rely solely on message transmission time or distance between nodes. However, it was shown that secure protocols utilizing both distance and location are possible. Their model specifically excluded protocols that rely on cooperation with other nodes, and hence their result does not directly apply to our protocol.

Eschenauer and Gligor [9] presented a simple key pre-distribution scheme based on randomly assigning a small set of keys (drawn from a larger pool) to each node, and Chan, Perrig, and Song [10] have extended this approach. Others have considered combinatorial key pre-distribution, such as Çamtepe and Yener [11] or Lee and Stinson [12], with the latter being utilized in this paper. In these deterministic schemes, a node's identifier determines which keys it holds via a public function, although the keys themselves are still chosen randomly. More specifically, the combinatorial design specifies the labels of the keys each node holds, but not the keys themselves.

Distance bounding protocols [13], utilize characteristics of the communication medium to put an upper bound on the distance between two nodes. Rasmussen and Čapkun [14] have recently demonstrated that extremely accurate distance bounding is possible over radio frequency. Their solution is practical for sensor networks.

A recent direction in sensor network research is to consider the use of directional antennas instead of omnidirectional antennas. Boudour et al. [15] have investigated how modern protocols can be changed to accommodate the use of directional antennas, and Ash and Potter [16] have demonstrated a method for sensor network localization that in some cases can estimate the location of a node to within 1 or 2 meters using directional antennas.

B. Organization

We describe our problem setting in more detail in Section II, followed by an explanation of the tools used by our network discovery protocol in Section III. Our method for secure network discovery is then presented in several parts Section IV. A basic analysis of the practicality of our solution is given in Section V, followed by some discussion of future work and concluding remarks in Section VI.

II. Problem Setting

Let $\mathcal{N} = \{N_1, N_2, \ldots, N_n\}$ be a collection of sensor nodes. Our focus is limited to *homogeneous* sensor networks, therefore the storage, computation, and communication capabilities of sensor nodes are identical. The method of communication used by all nodes will be omnidirectional wireless broadcasts. More specifically, all nodes have a fixed communication radius d such that any message sent by a given node will be overheard by any node within distance d and no others. The set of nodes within distance d of N_i will be referred to as N_i's *neighborhood*, with the nodes in N_i's neighborhood being referred to as N_i's *neighbors*.

In order to facilitate secure communication between sensor nodes, a *key pre-distribution scheme (KPS)* will be utilized to issue each node with a subset of cryptographic keys selected from a master key list. The set of keys issued to each node will be determined by its unique identity, which for simplicity will be represented by N_i for node i. Although nodes are free to perform both unencrypted or encrypted communication, transmission of sensor readings should only be done if a secure path exists all the way from the source to the destination. By secure, we mean that each pair of nodes communication along the path must be able to communicate under a shared key.

Upon deployment, nodes are distributed geographically randomly (i.e. a *fixed, no control network*). Thus, upon activation no node has any knowledge of its neighbors. The goal of each node is to discover the local network topology to the extent that it can establish multiple disjoint paths between itself and a base station or other desired destination. We require multiple disjoint paths rather than the shortest path so that perfectly secure message transmission can be utilized. This ensures that a node can always report its readings even if in the presence of an adversary, assuming the assumptions of the protocol are satisfied.

The process of deploying the sensor network (say, for example, by dropping them out of a plane) may cause nodes to become damaged or to malfunction. Thus, each node will only route messages through those nodes it can be convinced are functioning properly and are capable of secure communication with others. Furthermore, if nodes are deployed in a hostile environment, then an adversary may attempt to reprogram a subset of the nodes, jam communication between nodes, or insert fake nodes into the network, thus providing additional incentives for each node to verify the correct functioning of those nodes it communicates with.

III. Tools and Assumptions

A. Authentication

Consider the simple problem of two nodes wishing to verify whether or not they both possess a shared key, and hence, are capable of secure communication with each other. One may attempt to use a simple challenge response protocol to solve this problem. N_i sends an encrypted message to N_j. If N_j is in possession of the key used to encrypt the

$$\begin{array}{lcl}
\text{(A)lice} & & \text{(B)ob} \\
\text{Pick } N_A \in_U \{0,1\}^k & & \\
 & \xrightarrow{A, N_A} & \textbf{if } fp_{sig} \approx fp_A \textbf{ cont} \\
 & & \text{Pick } N_B \in_U \{0,1\}^k \\
\textbf{if } fp_{sig} \approx fp_B \textbf{ cont} & \xleftarrow[]{\substack{B, N_A, N_B \\ MAC_K(B, N_A, N_B)}} & \\
A \text{ accepts } B & & \\
 & \xrightarrow{\substack{A, N_A, N_B \\ MAC_K(A, N_A, N_B)}} & \textbf{if } fp_{sig} \approx fp_A \textbf{ cont} \\
 & & B \text{ accepts } A
\end{array}$$

Figure 1. The Fingerprinted Mutual Authentication Protocol (FMAP) [17]. Here fp_{sig} denotes the fingerprint of the received message, fp_A and fp_B the fingerprints of Alice and Bob respectively, and K is a key shared by Alice and Bob.

message, then it can respond with the correct plaintext message. Unfortunately, such a simple solution is not sufficient. Consider a node N_i attempting to verify a shared key K with a malicious node M_j who does not actually possess the key. A malicious node M_k who does possess the key could reply in place of M_j, thereby spoofing M_j's identity. For this reason, we introduce the following assumption:

Assumption 1. ***(Identity)*** *Each node has the ability to distinguish between two messages from the same source, and two messages from different sources.*

This assumption allows N_i to verify that the response to its challenge came from M_j and no one else. In practice, it can be realized through the methods described in [17], [18], which provide a mechanism for obtaining a fingerprint unique to each node from a received RF signal. Moreover, a mutual authentication protocol utilizing fingerprinting is given in [17] which we will use in our protocol. This protocol is presented in Figure 1. No formal proof of security is provided, but the protocol is similar in spirit to the secure SKID3 protocol [19] (Section 10.17) with additional checks added to make sure the fingerprints are as expected. We will henceforth refer to this as the *Fingerprinted Mutual Authentication Protocol (FMAP)*.

The FMAP protocol as presented assumes that each node is pre-loaded with the fingerprint of each other node. In order to avoid the overhead this would cause (and the inability to add nodes to the network at a later time), we have each node commit to its identity by broadcasting a simple `HELLO` message containing its claimed identity (and hence its key list). Each node that overhears such a message can verify that the fingerprint has not been seen before and store it locally. This allows us to make the following assumption:

Assumption 2. ***(Uniqueness)*** *Each node can commit to at most one unique identity.*

The node identity and uniqueness assumptions allow us to ignore the possibility of many spoofing attacks and the Sybil attack [20].

It is important to note that the FMAP protocol only demonstrates that node M_j has the ability to encrypt and decrypt messages under a given key K. This does not exclude the case where M_j may use M_k as an encryption or decryption oracle. More specifically, M_j may forward the challenge message to a colluding node M_k, who then responds to M_j with the appropriate response, who then replies to N_i. Because N_i receives the response from the "correct" node, the fingerprints will match and N_i will be falsely convinced that M_j possesses a shared key. In order to combat this we reiterate an assumption from Section II:

Assumption 3. ***(Uniformity)*** *All nodes have uniform communication range and all messages sent by a node are received by all of its neighbors.*

By this assumption, if M_j attempts to forward the challenge to a colluding node, then N_i will overhear it. Furthermore, any malicious neighbor of N_i cannot send a message M_j without N_i overhearing it. In practice, this requires silence from other nodes in N_i's neighborhood until the mutual authentication protocol completes. Should this be violated, the protocol will have to be re-run.

Theorem 1. *If two nodes successfully complete the FMAP protocol using key K, then both nodes must be in possession of K and be within each other's neighborhoods, except with very small probability.*

Proof: The identity, uniqueness, and uniformity assumptions combined ensure that the protocol can only be successfully completed if both nodes know K, or can guess it, and if the fingerprints on each message are as expected. Assuming the underlying mutual authentication protocol is secure, an adversary is only successful if it can guess K and can successfully spoof fingerprints. In [18] an error rate as low as 0.24% on fingerprint recognition is claimed. ■

Some problem settings consider nodes that can lower their communication range in an effort to reduce power consumption, thus violating the uniformity assumption. If nodes were allowed to selectively reduce their communication range then two nearby malicious nodes may be able to exchange messages without detection by others. This would prevent us from proving Theorem 1.

B. Key Pre-Distribution

We established in the previous section that a node can recognize two messages from the same source, that a node can commit to at most one identity, and that two nodes can verify a shared key. This does not, however, allow us to actually verify the identity (and key list) of a node. Because a given node will be relying on its shared-key neighbors to forward messages to those node with which it does not share a key, and also to nodes outside of its communication range, it is desirable for a node to be able to prove its identity (and therefore its key list), thus proving who it can communicate with securely. To aid in solving this problem, we use a key pre-distribution scheme (KPS) that allows for a node to be uniquely identified by the keys that it possesses. For example, the LS-KPS scheme of Lee and Stinson [12] allows a node to be uniquely identified by knowledge of any two keys that it possesses, while maintaining practical system parameters. Although a general scheme is presented in [12], we focus on two specific variants: The *linear* LS-KPS, which is extremely efficient, and the *multiple space* MS-KPS, which sacrifices a small amount of efficiency for a boost in resilience.

The Linear LS-KPS: For a network of size n. let k be the number keys issued to each node, let r be the number of nodes possessing a given key, and let v be the total size of the key pool. A linear $(v, n, r, k; \mu)$-LS-KPS can be derived for certain values of v, n, r, and k. More specifically, let p be a prime power such that $2 \leq k \leq p$. Then it is shown in [12] that a (kp, p^2, p, k)-LS-KPS can be constructed. To construct such a scheme, let $X \subseteq \mathbb{F}_p$ such that $|X| = k$. Each node will be issued a label (identifier) $(a, b) \in \mathbb{F}_p \times \mathbb{F}_p$ which determines the set of key identifiers $A_{a,b} = \{(x, ax + b) \,:\, x \in X\}$.

It is shown in [12] that two nodes $N_{a,b}$ and $N_{a',b'}$ can determine if they share a key through the following:

1) If $a = a'$ then $N_{a,b}$ and $N_{a',b'}$ do not share a common key.
2) If $a \neq a'$ then compute $x = (b'-b)(a'-a)^{-1} \mod p$. If $0 \leq x \leq k-1$ then $N_{a,b}$ and $N_{a',b'}$ share the common key $K_{x,ax+b}$. Otherwise $N_{a,b}$ and $N_{a',b'}$ do not share a common key.

As an example of the linear LS-KPS, let $p = 47$ and $k = 30$. Then $v = kp = 1410$, $n = p^2 = 2209$ and a linear $(1410, 2209, 47, 30)$-LS-KPS can be constructed. Thus, if each node stores just 30 keys, then we can accommodate up to 2209 nodes while maintaining the property that a node is uniquely identified by any two keys it possesses.

The LS-KPS is well-suited to sensor networks as two nodes can determine whether or not they share a key using only two integer subtractions and one inverse modulo p, where p is very small.

The Multiple Space MS-KPS: The linear LS-KPS can be used to distribute shares of a Blom scheme [21], rather than the indexes of a key. In the multiple space scheme, the storage requirement of each node is doubled, and the cost of key computation is increased by a constant number of additions and multiplications to compute the key from the Blom scheme. The benefit to such an approach is a significant increase in resilience to key compromise. As shown in [12], the multiple space LS-KPS decreases the probability of a compromised affecting other links in the network by an order of magnitude, and often more when only a small fraction of nodes are compromised. Note that this analysis assumed that the memory size of each node is fixed, so a single node may not possess more keys at a time than it was originally issued.

Despite the fact that no two nodes ever have more than one key in common the LS-KPS can achieve very high levels of connectivity and resilience. Setting $k = 50$ and $p = 149$, the left table shows the probability that a one or two hop path exists between two nodes N_i and N_j when η nodes lie in their common neighborhood (i.e., are candidates to route messages between them), and the right table shows the probability that the link between them is compromised given that the adversary has learned keys from s nodes. In the left table, $\mathbf{Pr}_1$, the probability of a 1-hop path existing, is a constant $\mathbf{Pr}_1 = \frac{1}{3}$.

η	$\mathbf{Pr}_2$	$\mathbf{Pr}_1 + \mathbf{Pr}_2$
1	0.074	0.407
2	0.139	0.472
3	0.197	0.530
4	0.249	0.582
5	0.295	0.628
10	0.460	0.793
15	0.551	0.885
20	0.602	0.936

s	LS-KPS	MS-KPS
2	0.0132	0.0001
5	0.0327	0.0017
10	0.0642	0.0073
20	0.1244	0.0281
30	0.1807	0.0590
40	0.2333	0.0972
50	0.2827	0.1404

C. Localization and Directional Antennas

Most sensor networks feature nodes that broadcast and receive messages using omnidirectional antennas, with their communication region being a sphere of radius d centered around the node. A more recent area of study has been the investigation of sensor networks utilizing only directional antennas that instead broadcast in a conical region. Although we do not consider this problem setting, there are tools that have been developed using directional antennas that are of interest, the most relevant being the localization methods by Ash and Potter [16]. Utilizing received signal strength and angle of arrival at multiple directional antennas on a single node, a node is able to estimate the position of a neighbor to an accuracy of less than 1m in some cases, with the ability to estimate the angle of arrival of an incoming signal to within $3°$. The methods in [16] are of particular interest as they do not rely on any trusted *anchor nodes* possessing additional hardware, such as a GPS device. Thus, their localization method is referred to as *self-localization*, as it does require any additional trust between nodes.

Accurate localization methods for sensor networks are still an active area of research. In the protocols presented below we will, when appropriate, assume that a node can use localization to get an estimate of a neighbor's location to within an error of ϵ, where ϵ is significantly smaller than the communication range d of each node.

IV. Proposed Solution

A. Phase 1: Identifying Neighbors

The first goal of each node is to discover which nodes are within its neighborhood and to verify their identity. In order to achieve this, each node will commit to an identity and then perform the FMAP protocol described in Section III-A with each neighbor it is supposed to share a key with. Each node then broadcasts the result of the FMAP protocol to all other nodes in their neighborhood so that they may decide whether or not to trust each node. For simplicity, we assume all nodes come online at the same time. Adding nodes into an existing network will be discussed later. Let N_i be the node running the protocol.

1) Broadcast a `HELLO` message containing the identity of node N_i.
2) For each `HELLO` message received, record the fingerprint and identity, making sure that the fingerprint has not already been seen.
3) For each node N_j with which a key is supposed to be shared with node N_i, broadcast (`BEGIN FMAP`, N_i, N_j) and run the FMAP protocol to verify the key.
4) a) If FMAP is successful, broadcast `ACCEPT KEY` N_j.
   b) If FMAP is not successful, broadcast `REJECT KEY` N_j.
5) a) If a neighbor N_j of N_i begins the FMAP protocol with N_k, record the tuple (N_j, N_k).
   b) If N_j broadcasts `ACCEPT KEY` N_k, verify that the tuple (N_j, N_k) has been recorded, and if so, add `ACCEPT KEY` to the tuple.
   c) If N_j broadcasts `REJECT KEY` N_k, verify that the tuple (N_j, N_k) has been recorded, and if so, add `REJECT KEY` to the tuple.
6) Wait until no new nodes have come online and no new `ACCEPT KEY` or `REJECT KEY` votes have been heard for a period of time t_0.
7) Mark any run of FMAP that has yet to complete as `REJECT KEY`.
8) Apply the "accept rule" (discussed below) to each N_j and broadcast the result as `ACCEPT ID` N_j or `REJECT ID` N_j.

At the end of this protocol, each node N_i has a list of those nodes in its neighborhood (both identity and fingerprint), has verified those nodes with which it shares a key, and knows the result of all other FMAP protocols run by its neighbors. It remains to be decided when a node should accept the identity of each of its neighbors (i.e., what the "accept rule" is). The parameter t_0 represents a timeout value, meant to determine whether or not nodes in the neighborhood are still in the process of completing their runs of the FMAP protocol.

Recall that the linear LS-KPS allows one node N_i to uniquely identify any other node N_j if it can be convinced of any two keys that N_j possesses. A consequence of this is that each node can verify at most a single key with any other node, for if the nodes had two keys in common this would imply that they were the same node. Similarly, if a malicious node attempts to lie about its identity then it possesses at most one key associated with the false identity it has assumed.

Definition 1. *A set of nodes casting a vote for N_j are said to form a t-Local Honest Majority (t-LHM) if:*

1) *There are more `ACCEPT KEY` votes than `REJECT KEY` votes;*
2) *at least t distinct keys were verified by nodes casting votes; and*
3) *if $\mathcal{K} = \{k_1, \ldots, k_m\}$ are the different keys verified with N_j using FMAP, then the previous two properties are satisfied for the set $\mathcal{K} - k_i$ for all $k_i \in \mathcal{K}$.*

In other words, a set of voting nodes forms a t-LHM if a majority of the nodes have cast an `ACCEPT KEY` vote over t different keys, even when all nodes possessing any single key are removed from the vote. This is intended to capture the fact that if a malicious node attempts to lie about its identity, then it can have at most one key consistent with that identity. Therefore, it can falsely convince all honest nodes possessing that one key to accept its false identity.

Theorem 2. *A node can cast at most one `ACCEPT KEY` or `REJECT KEY` vote for another node. Furthermore, a node can only vote for another node with which it supposed to share a key and with which it has run the FMAP protocol.*

Proof: In Step 3 of the protocol, each node broadcasts the fact that it is beginning the FMAP protocol with another node. Each node that overhears a run of the protocol records this fact in the tuple (N_j, N_k). Because the list of keys given to each node is derived from its identity, it can easily be verified that N_j and N_k are supposed to share a key. By the uniformity assumption, if both N_j and N_k are in a node's neighborhood, then it will overhear each step of the FMAP protocol between them. Thus, only those tuples/votes that are valid with respect to the previous observations will be used to establish a node's identity. The fact that a node may cast at most one vote follows directly from the uniqueness and identity assumptions. ■

Note that Theorem 2 does not guarantee that the FMAP protocol was successful or not between nodes N_j and N_k, only that N_i can observe that the appropriate messages were exchanged and that only one vote was cast by each. It is possible that two malicious nodes, one or both lying about their identities, could fake the FMAP protocol and cast a false `ACCEPT KEY` vote for each other. Similarly, a malicious node may successfully complete the FMAP protocol with another but vote `REJECT KEY` anyways. A malicious node could also announce it was beginning the protocol with a node that it cannot talk to, and then announce the result. This is undetectable by N_i, but will be addressed in the next phase of the protocol.

Definition 2. *The t-LHM-Accept Rule:*

1) *If N_i is supposed to share a key with N_j and does not complete the FMAP protocol with N_j, then reject N_j.*
2) *If the set of nodes casting a vote for N_j form a t-LHM, then accept the identity of N_j as valid;*
3) *otherwise, reject the identity of N_j.*

The t-LHM-Accept Rule relies on the fact that if there enough honest nodes in N_i's neighborhood, then they can always achieve a majority when voting on N_j's honesty. In order to prove the security of the protocol we must put a bound on the number of malicious nodes participating in a given vote.

Assumption 4. ***(LHM)*** *Let $\mathcal{K}$ be the set of keys possessed by N_j that are also possessed by common neighbors of N_i and N_j (including the shared key between N_i and N_j if it exists). Then $|\mathcal{K}| \geq 3$ and there are more honest nodes in the common neighborhood possessing keys from the set $\mathcal{K} - k_i$ for all $k_i \in \mathcal{K}$ than malicious nodes.*

This assumption is intended to capture a necessary condition for the 2-LHM-Accept Rule to also be a sufficient condition for determining a neighbor's honesty.

Theorem 3. *If a node N_j is accepted by 2-LHM-Accept and the LHM assumption holds, then N_j is not lying about its identity.*

Proof: We first note that if N_j fails the FMAP protocol with N_i then it is immediately rejected by N_i. Assume N_j is lying about its identity, then N_j has at most one key K consistent with its claimed identity. Also, recall that, by Theorem 2, each node can cast at most one vote. The 2-LHM property will only be satisfied if the number of `ACCEPT KEY` votes from nodes not possessing K is greater than the number of `REJECT KEY` votes, and at least 2 other keys are verified. Each honest node that does not possess K will cast one `REJECT KEY` vote for N_j, and in the worst case each malicious node not possessing K will cast at most one `ACCEPT KEY` vote. By the LHM assumption, the number of `REJECT KEY` votes is greater than the number of `ACCEPT KEY` votes among neighbors not possessing K. Hence, the 2-LHM-Accept rule is not satisfied and N_j's identity will not be accepted. ■

The LHM assumption also places a limit on the capability of malicious nodes to prevent honest nodes from being accepted.

Theorem 4. *If the LHM assumption holds then a coalition of malicious nodes cannot force an honest node N_j to be rejected by N_i with the 2-LHM-Accept Rule.*

Proof: If an honest node N_j is rejected, then it must have received more `REJECT KEY` votes than `ACCEPT KEY` votes. Recall that, by Theorem 2, each node can cast at most one vote. Because N_j is honest, each `REJECT KEY` vote must have come from a distinct malicious node. The LHM assumption guarantees that there are always more honest nodes than malicious nodes in situations considered by the 2-LHM-Accept Rule, and hence, there can never be more `REJECT KEY` votes than `ACCEPT KEY` votes for an honest N_j. ■

Theorems 3 and 4 together prove the correctness of our neighbor discovery protocol. It is worth noting that if N_i completes the FMAP protocol with N_j, then N_i has directly verified that N_i possesses a single key K consistent with its claimed identity. This means the 1-LHM-Accept Rule is sufficient in the case where N_i and N_j share a key, with the caveat that 1-LHM must verify a key different from K.

B. Phase 2: Identifying 2-Hop Paths

At the conclusion of Phase 1, each node has a list of its neighbors and has decided whether or not to accept their claimed identities. From this point on, we assume that an honest node N_i will ignore the existence of any node N_j it could not verify during Phase 1. To simplify presentation we also assume that N_i can establish a path to any neighbor it does not share a key with via its local neighbors. The table in Section III-B shows the probability that 2-hop paths exist to any neighbor, and longer length paths would be possible to calculate from N_i's verified neighbors, thereby increasing the likelihood a local path can be established. Our goal in this section is to establish 2-hop neighbors that exist outside of N_i's neighborhood (i.e., paths of the form $N_i \rightarrow N_j \rightarrow N_k$).

Verifying the identity of N_i's 2-hop neighbors is different than identifying nodes in N_i's neighborhood, as we lose the ability to verify fingerprints on N_k, as well as the ability to overhear whether or not the FMAP protocol was actually executed. Furthermore, we cannot assume that an intermediate node N_j is honest, only that it is not lying about its identity. Recall that in Step 5a of Phase 1, node N_i records the tuple (N_j, N_k) when the FMAP protocol is initiated by a neighbor. If N_k is also in N_i's neighborhood, then the corresponding tuple (N_k, N_j) is also recorded. Thus, during Phase 1, node N_i also learns which nodes N_k lie outside of its neighborhood such that N_j has successfully completed the FMAP protocol with N_k, and whether or not they were accepted in Step 8. Such nodes are referred to as *2-hop neighbors*. We can use this information to apply a voting protocol similar to Phase 1.

Our main problem during this phase is identifying those nodes that are attempting to vote on the identity of N_k, but who are not actually within N_k's communication range. Let $\mathcal{N}_i$ denote N_i's communication range (i.e., the nodes in N_i's neighborhood). Any node in $\mathcal{N}_i$ can cast a vote for N_k, but only those nodes that lie in the region $\mathcal{N}_i \cap \mathcal{N}_k$ are capable of routing a message between N_i and N_k. Nodes

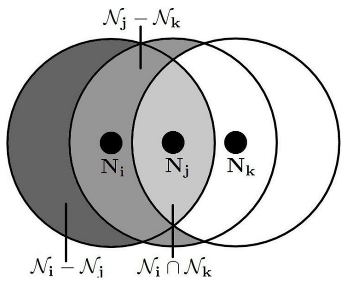
$\mathcal{N}_j - \mathcal{N}_k$
N_i
N_j
N_k
$\mathcal{N}_i - \mathcal{N}_j$
$\mathcal{N}_i \cap \mathcal{N}_k$

Figure 2. Only nodes in $\mathcal{N}_i \cap \mathcal{N}_k$ are capable of routing a message, but malicious nodes in $\mathcal{N}_j - \mathcal{N}_k$ learn about N_k from N_j, and nodes in $\mathcal{N}_i - \mathcal{N}_j$ learn about N_k from malicious nodes in $\mathcal{N}_j - \mathcal{N}_k$, thus enabling them to falsely claim they can route to N_k.

in the region $\mathcal{N}_j - \mathcal{N}_k$ are unable to route a message to N_k, but learn of N_k via N_j during Phase 1. Nodes in the region $\mathcal{N}_i - \mathcal{N}_j$ are unable to route a message to N_k, and do not overhear N_j during Phase 1 (but could overhear a node in $\mathcal{N}_j - \mathcal{N}_k$ that repeats the fact). This means that a malicious node in $\mathcal{N}_j - \mathcal{N}_k$ could falsely broadcast that it is beginning the FMAP protocol with N_k and then cast a vote. Similarly, if a node in $\mathcal{N}_j - \mathcal{N}_k$ falsely begins the protocol, then a malicious node in $\mathcal{N}_i - \mathcal{N}_j$ learns of N_k and can do the same. The fact that any of N_i's neighbors is capable of learning of N_k's existence means that in the worst case every malicious node in N_i's neighborhood that shares a key with N_k will cast a false vote.

Clearly, if there are more honest nodes in $\mathcal{N}_i \cap \mathcal{N}_k$ that can route messages between N_i and N_k than malicious nodes in $\mathcal{N}_i$, then the malicious nodes cannot cast enough `REJECT ID` votes to prevent N_k from being accepted. Unfortunately, the malicious nodes can still falsely accept N_k's identity, thus suggesting a secure path exists where it actually does not. For this reason, we have each N_j include localization information about N_k, allowing malicious nodes to be detected in many cases.

Assumption 5. *The* Honest 2-Hop Majority (H2M) *assumption states that there are more honest nodes in $\mathcal{N}_i \cap \mathcal{N}_k$ that can route messages between N_i and N_k than malicious nodes in $\mathcal{N}_i$.*

Let the protocol in Phase 1 be altered such that each node includes information on the location of N_j (i.e., the relative angle and distance) when it broadcasts its `ACCEPT ID` vote. Furthermore, we assume that N_i can compute similar localization information on N_j as needed. Then the protocol for identifying 2-hop neighbors from N_i's point of view is:

For each 2-hop neighbor N_k:

1) For each accepted neighbor N_j casting an `ACCEPT ID` vote for N_k
   a) Compute $(N_j, N_k,$`ACCEPT ID`$,\text{loc}(N_k))$, where loc(N_k) is the location of N_k relative to N_i.
2) For each neighbor N_j casting a `REJECT ID` vote for N_k
   a) Compute $(N_j, N_k,$`REJECT ID`$,\text{loc}(N_k))$, where loc(N_k) is the location of N_k relative to N_i.
3) Wait until no new votes for N_k have been received for a period of time t_1.
4) Let $\mathcal{S}_k$ be the largest subset of tuples from Steps 1 and 2 such that all estimates of loc(N_k) are with 2ϵ of each other and $N_j \in \mathcal{N}_i \cap \mathcal{N}_k$ (according to the consensus on N_k's location), where ϵ is the maximum error of localization.
5) a) If there are more `ACCEPT ID` votes in $\mathcal{S}_k$ than `REJECT ID` votes, then accept all N_j in $\mathcal{S}_k$ who cast an `ACCEPT ID` vote as being able to route messages to N_k. Broadcast the message `ACCEPT ROUTE` N_j, N_k.
   b) If there are more `REJECT ID` votes in $\mathcal{S}_k$ than `REJECT ID` votes, then do not accept N_k as being an honest node and do not include them in any path.

As with the protocol in Phase 1, this protocol relies on the fact that a majority of the nodes voting on N_k are honest. If this assumption holds, then there will exist a majority of nodes that agree on N_k's position, who also lie within communication range of N_i and N_j. The parameter t_1 is a timeout value used to ensure all votes for N_k have been received before proceeding.

Theorem 5. *If the H2M assumption holds, and if $N_j \notin \mathcal{N}_i \cap \mathcal{N}_j$, then $N_i \to N_j \to N_k$ will not be accepted as a valid 2-hop path, except with small probability.*

Proof: In Step 4 of the protocol, $\mathcal{S}_k$ is chosen to be the largest set of N_j such that each N_j agrees on N_k's position, and lies in the intersection of N_i and N_k's communication range. By the H2M assumption, the honest N_j's will agree on N_k's position with accuracy 2ϵ, thus defining the common intersection of N_i and N_k's communication range to within 2ϵ. Hence, any node lying about its ability to communicate with N_k must be within 3ϵ of N_k's communication range, as we can only estimate N_j's position with accuracy ϵ. ■

Theorem 5 allows us to conclude with high probability that only votes from those nodes that are actually capable of communication with N_k will included in the set $\mathcal{S}_k$.

Theorem 6. *If the LHM and 2HM assumptions are satisfied, and N_k is accepted by the protocol, then N_k exists and is not lying about its identity.*

Proof: This follows from Theorem 3 and the fact that by the 2HM assumption a majority of the N_j in $\mathcal{S}_k$ are honest. If N_k is lying about its identity, then each honest N_j will have detected this in Phase 1 and broadcasted a reject vote.

Hence, N_k has proven its identity to each honest N_j in $\mathcal{S}_k$. ■

Note that during Phase 2, a node discovers if any malicious node attempted to claim it could route messages to N_k when it was not in communication range. Thus, we can mark such nodes as untrusted and not consider any path containing them in the future.

C. Phase 3: Beyond 2-Hop Paths

We now consider how a node can extend its knowledge of the network to longer paths. At the end of Phase 2, each node has determined its 2-hop neighbors and broadcasts that fact publicly. Additionally, any malicious node that has claimed a false identity, or the ability to communicate with a 2-hop neighbor when it cannot, has been discovered, except with small probability. We continue from this point assuming that each node has established a path to each of its neighbors, as well as to its 2-hop neighbors, and that each of these paths are valid.

In Phase 1, each N_i learns paths of the form $N_i \to N_j$, and in Phase 2 each N_i learns paths of the form $N_i \to N_j \to N_k$. Just as nodes informed their neighbors of the results of Phase 1 so that the information could be utilized to construct 2-hop paths, each node broadcasts the results of Phase 2 so that nodes if their neighborhood learn which 3-hop paths exist. More specifically, each N_j will broadcast all paths it has discovered of the form $N_j \to N_k \to N_l$.

Recall that each N_j accepted in Phase 2 must have proven they are capable of communication with N_k (by proving their identity in Phase 1), that they lie within N_k's communication range, and also that the N_j's form an honest majority. Therefore, if a majority of the nodes N_j that are capable of communicating with N_k broadcast knowledge of a route N_l via N_k, then N_i can conclude that said route does in fact exist. N_i's view of the network can be updated accordingly, and N_i can inform its neighbors of the fact that a secure path to N_l has been determined.

The same process can be followed for routes of arbitrary length. During each subsequent phase N_i increases its knowledge of the network by one hop by relying on the nodes that were verified during Phase 1 and 2 of the protocol. In practice, this process would be repeated until each sensor node has discovered a base station, the desired destination node, or for a fixed maximum number of hops to ensure it eventually terminates.

D. Adding New Nodes

In order to accommodate new nodes being added to the network, we observe that once network discovery has completed, each node has established its own view of the network. When a new node comes online, it can announce its presence through a `HELLO` message and engage in the FMAP protocol with its neighbors to establish its identity. This is essentially re-running Phase 1 of the protocol, where, upon completion, each node in the neighborhood will accept or reject the new node's identity. By broadcasting these votes, surrounding nodes are informed of the new node and its status, and the information is propagated throughout the network as each node repeats whether or not it accepts or rejects the new node, as described in Phase 3. The new node can learn about the rest of the network by querying its neighbors. Because an honest majority is assumed, the new node will accept a path through the network as valid only if a majority of its neighbors agree on its existence.

V. Performance Analysis

Our protocol relies heavily on the assumption that there always exists an honest majority that can be trusted. In order to justify that such an assumption is practical, we investigate how likely it is that our LHM and 2HM assumptions are satisfied as more malicious nodes are added to a network. To do so, we run a Java simulation where 50 keys are assigned to 2000 randomly labeled nodes with communication radius 1 using a $(50 \cdot 149, 149^2, 149, 50)$-LS-KPS as described in Section III-B, which are then distributed randomly over a square region according to a density parameter δ, such that we expect $\pi\delta$ nodes to lie in any circle of radius 1. Note that nodes around the perimeter of this area will have fewer nodes in their neighborhood than interior nodes on average. Our simulation includes results from all nodes in the network, including perimeter nodes.

To test the LHM assumption we take every pair of neighbors (N_i, N_j) and test whether or not the nodes in N_i's neighborhood satisfy the LHM assumption with respect to N_j as a growing number of malicious nodes are added to the network. The left graph in Figure 3 shows our results. We observe that at low network densities the LHM assumption is difficult to satisfy due to the fact that among N_i and its neighbors there must be at least three distinct keys shared with N_j. Once network density is sufficiently high, we see that the LHM assumption is satisfied nearly 100% of the time, even when 100+ nodes present during network discovery are malicious.

To test the 2HM assumption we take every pair of 2-hop neighbors (N_i, N_k) such that there exist at least three different N_j capable of routing a message between N_i and N_k, and test whether or not the 2HM assumption is satisfied with respect to N_i and the set of N_j nodes as a growing number of malicious nodes are added to the network. We limit our analysis to situations where three or mode intermediate nodes exist because voting on the validity of a link does not make sense until at least three nodes are involved. The right graph in Figure 3 shows our results. We observe that at low node density the 2HM assumption is satisfied with higher probability. This is due to the fact that situations where three intermediate nodes exist are somewhat rare, and a larger number of malicious nodes are required before it is expected that one lies within any given node's

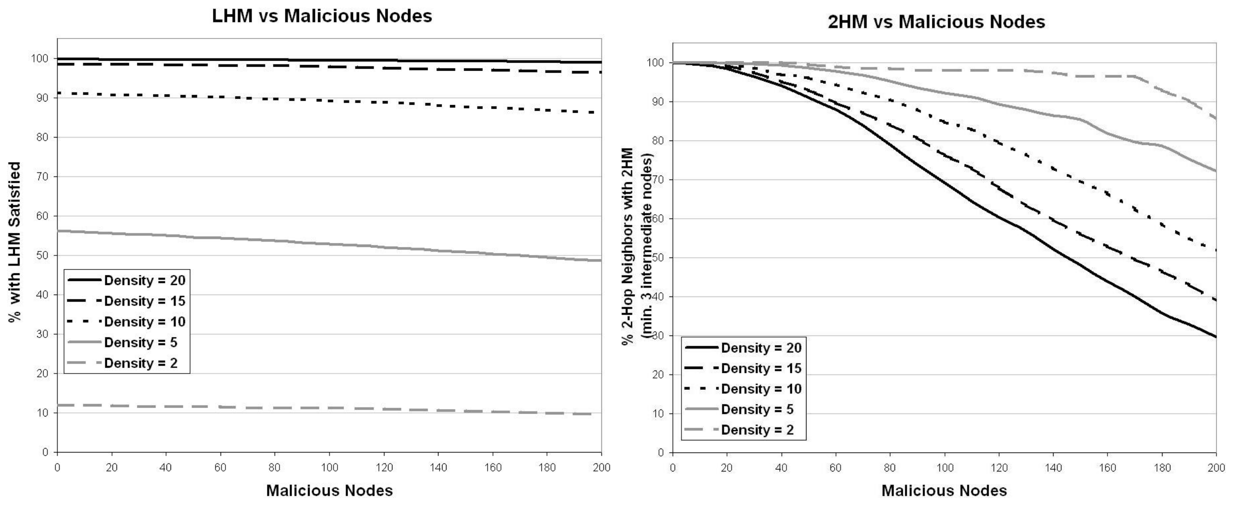

Figure 3. The probability that the LHM and 2HM assumptions are satisfied for a random network of 2000 nodes, each possessing 50 keys, for varying numbers of malicious nodes. The left graph considers all pairs of neighbors, while the right graph only considers pairs of 2-hop neighbors such that at least three nodes exist that can route messages between them.

communication range. At higher densities we observe that the 2HM is still satisfied with high probability even when 50+ nodes are malicious.

In order to study the effect of parameter choice on the probability that the LHM and 2HM assumptions are satisfied, we repeated the previous experiments for a variety of different parameter sets. The results are summarized in Figure 4 for the fixed density $\delta = 20$. We observe that performance is closely related to the ratio between k and p, which is reflected in the network itself by the probability of two nodes sharing a kew increasing as the ratio of k to p approaches 1. When connectivity is high, more malicious nodes must be present before the LHM or 2HM assumptions are violated.

VI. Conclusion

We have considered a solution to the problem of discovering disjoint paths in a sensor network that assumes an active adversary is present in the network from the very beginning. Although several assumptions are required to ensure the protocol is secure, we demonstrate that practical tools, such as combinatorial key pre-distribution, fingerprinting, and self-localization, can be used to realize these assumptions. We also show that the resulting protocol is resilient in the presence of many adversarial nodes. Because we rely on fingerprinting and localization, there always exists a small probability that a malicious node could exploit the inherent error in these techniques, however, we emphasize that our goal was to establish disjoint paths so that perfectly secure message transmission protocols can be utilized. Such protocols are designed assuming an adversary controls up to one third of the paths between nodes.

It may be possible to strengthen our protocol by performing additional analysis on the votes received in Phase 1 and 2. A `REJECT` vote implies that at least one of the voters is malicious, but determining which node(s) solely from the votes overheard among nodes would allow us to detect malicious nodes in some situations where they are not lying about their identity. It may also be possible to relax our need of localization in Phase 2 and instead rely only distance bounds or angle of arrival information. We leave these problems as future work.

Acknowledgment

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through the grant NSERC-RGPIN #203114-06.

References

[1] K. M. Martin and M. Paterson, “An application-oriented framework for wireless sensor network key establishment,” *Electron. Notes Theor. Comput. Sci.*, vol. 192, no. 2, pp. 31–41, 2008.

[2] D. Dolev, C. Dwork, O. Waarts, and M. Yung, “Perfectly secure message transmission,” *J. ACM*, vol. 40, no. 1, pp. 17–47, 1993.

[3] J. Wu and D. R. Stinson, “Three improved algorithms for multi-path key establishment in sensor networks using protocols for secure message transmission,” *IEEE Transactions on Dependable and Secure Computing*, vol. 99, 2010.

[4] C. Karlof and D. Wagner, “Secure routing in wireless sensor networks: attacks and countermeasures,” *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 293–315, 2003.

[5] K. Sanzgiri, B. Dahill, B. Levine, C. Shields, and E. Belding-Royer, “A secure routing protocol for ad hoc networks,” in *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*, 2002, pp. 78–87.

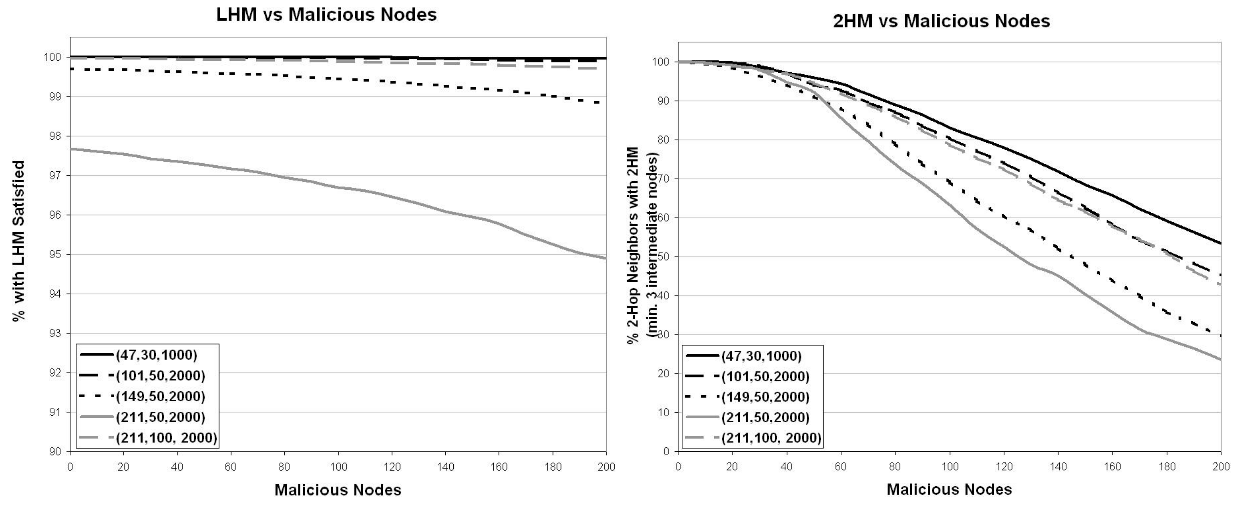

Figure 4. The probability that the LHM and 2HM assumptions are satisfied for a variety of different parameter sets using a fixed density of 20. Here, each data set is marked with the tuple (p, k, n), where n is the number of nodes. The left graph considers all pairs of neighbors, while the right graph only considers pairs of 2-hop neighbors such that at least three nodes exist that can route messages between them.

[6] M. G. Zapata and N. Asokan, "Securing ad hoc routing protocols," in *WiSE '02: Proceedings of the 1st ACM workshop on Wireless security*. New York, NY, USA: ACM, 2002, pp. 1–10.

[7] P. Papadimitratos and Z. Haas, "Secure on-demand distance vector routing in ad hoc networks," in *Advances in Wired and Wireless Communication, 2005 IEEE/Sarnoff Symposium on*, apr. 2005, pp. 168 –171.

[8] M. Poturalski, P. Papadimitratos, and J.-P. Hubaux, "Secure neighbor discovery in wireless networks: formal investigation of possibility," in *ASIACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security*. New York, NY, USA: ACM, 2008, pp. 189–200.

[9] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*. ACM Press, 2002, pp. 41–47.

[10] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *IEEE Symposium On Security and Privacy*, 2003, pp. 197–213.

[11] S. A. Çamtepe and B. Yener, "Combinatorial design of key distribution mechanisms for wireless sensor networks," in *9th European Symposium On Research Computer Security*, 2004, pp. 293–308.

[12] J. Lee and D. R. Stinson, "On the construction of practical key predistribution schemes for distributed sensor networks using combinatorial designs," *ACM Trans. Inf. Syst. Secur.*, vol. 11, no. 2, pp. 1–35, 2008.

[13] S. Brands and D. Chaum, "Distance-bounding protocols (extended abstract)," in *EUROCRYPT '93, Lecture Notes in Computer Science 765*. Springer-Verlag, 1993, pp. 344–359.

[14] K. B. Rasmussen and S. Čapkun, "Realization of RF distance bounding," in *Proceedings of the USENIX Security Symposium*, 2010.

[15] G. Boudour, A. Lecointre, P. Berthou, D. Dragomirescu, and T. Gayraud, "On designing sensor networks with smart antennas," in *7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems*, Toulouse France, 2007, pp. p. 349–356.

[16] J. N. Ash and L. C. Potter, "Sensor network localization via received signal strength measurements with directional antennas," in *Proceedings of the 2004 Allerton Conference on Communication, Control, and Computing*, 2004, pp. 1861–1870.

[17] K. B. Rasmussen and S. Čapkun, "Implications of radio fingerprinting on the security of sensor networks," in *Proceedings of IEEE SecureComm*, sep. 2007, pp. 331 –340.

[18] B. Danev and S. Čapkun, "Transient-based identification of wireless sensor nodes," in *IPSN '09: Proceedings of the 8th IEEE/ACM Information Processing in Sensor Networks*. IEEE/ACM, 2009, pp. 25–36.

[19] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 2001.

[20] J. R. Douceur, "The sybil attack," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS '01. London, UK: Springer-Verlag, 2002, pp. 251–260.

[21] R. Blom, "Non-public key distribution," in *CRYPTO*, 1982, pp. 231–236.