9-9-2007

Cooperative strategies for pairwise secure communication channels in sensor networks

Peter Oelschlaeger

Cooperative Strategies for Pairwise Secure Communication Channels in Sensor Networks

by

Peter M. Oelschlaeger

B.S., Computer Science, Trinity University, 2001

THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

July, 2007

Acknowledgments

I would first like to thank the outstanding faculty and staff in the Department of Electrical and Computer Engineering and the Department of Computer Science at the University of New Mexico for their instruction, guidance, and assistance. Special thanks to my advisor, Dr. Greg Heileman, and my committee members, Dr. W. Wennie Shu and Dr. A. Barney Maccabe for their insight and constructive comments on my research.

I owe a great big heap of gratitude to my parents, Max and Mary Oelschlaeger, for their continuous assistance and encouragement. I also give a playful thanks to my dog, Paco, who has a knack for delivering timely, gentle reminders that today, despite what the conventional wisdom may say, is a great day indeed.

Finally, although it seems trite and belated, I owe thanks most of all to my wife, Sarah Flaig, for her unparalleled patience, love, and support.

Cooperative Strategies for Pairwise Secure Communication Channels in Sensor Networks

by

Peter M. Oelschlaeger

ABSTRACT OF THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

July, 2007

Cooperative Strategies for Pairwise Secure Communication Channels in Sensor Networks

by

Peter M. Oelschlaeger

B.S., Computer Science, Trinity University, 2001

M.S., Electrical Engineering, University of New Mexico, 2007

Abstract

Establishing secure communication channels in sensor networks is made especially difficult because of low power resources, hostile environments, and wireless communication. The power requirements of traditional cryptographic methods create the need for alternative strategies for secure communication in sensor networks. This thesis explores key distribution techniques in sensor networks. Specifically, we study in depth one method that enables sensors to establish pairwise secure communication channels. This strategy relies on a cooperative set of peer sensors to construct a unique key between two sensors. We built a unique network simulator to test secure communication parameters in a typical deployment scenario. This research tests the strategy by which the cooperative set of sensors is chosen. The results demonstrate that a strategy favoring neighbor nodes consumes significantly less energy than other alternatives at the expense of vulnerability to geographically localized attacks.

Contents

List of Figures

Chapter 1

Distributed Sensor Networks

A sensor network is a colony of loosely coupled, autonomous, stationary or mobile nodes. Thus far, the main thrust of research in sensor networking has been the design of suitable sensor nodes, delivery vehicles, and communication protocols. More recently, a variety of work has been published related to the specific requirements of securing sensor networks. Since these networks lack a physical infrastructure, power is at a premium, and computation intensive security measures are infeasible in most deployments. These networks present a paradox: the nodes (often called motes) must consume a minimum of power to preserve the lifespan of the network, yet a compromised network could be as useless as a dead one, or worse. The network must meet today's security requirements equipped with only yesterday's technology [4]. In defining sensor network security, we seek a compromise — one that will provide adequate security against most types of attacks without excessive drain on the power resources.

Any sensor network must address a few basic concerns. These concerns include organization, scalability, openness and authentication, fault tolerance, power and efficiency, deployment, and size. Whereas most traditional networks are concerned

with latency, bandwidth utilization, and fairness, sensor networks are more concerned with energy consumption and adaptability. The real issues in sensor net design are not the size nor shape of the node, whether it is stationary or mobile, or what kind of batteries it requires to run, but rather the methods of communication among the nodes and the organizational properties of their respective heterogeneous networks. Today, we already have realized a wide array of uses for sensor networking. Habitat monitoring, environmental control, and navigation scenarios are a few of the possible applications envisioned by sensor network designers. In each of these applications, we can see a need for security. A habitat monitor producing erroneous data is useless. A network directing traffic according to maliciously modified traffic could create massive havoc. However, the importance of security is even higher in military applications. As a longstanding prime benefactor of research in the United States, the military and the Department of Defense contribute heavily to sensor networking. Defending frontiers, monitoring potentially (or already) hostile scenarios, assisting in the coordination of the tactical battlefield, and protecting (or tracking) targets are the primary interests of the defense community. These types of uses can be categorized generally as perimeter security and we can easily extend our thinking about security, and the possible consequences of a breach in security to this domain [19]. Some of the distinctions are subtle; for instance, an attacker equipped with the knowledge that the U.S. military uses sensor networks to protect important assets, could deduce the presence of an important person by merely detecting the presence network traffic. In these scenarios, location information is critical. Extrapolating the precise whereabouts of sensor nodes could lead to a physically compromised network or reveal tactical strategy. Not only could the network be disabled, in some cases it could be turned against those who deployed it.

There exists a gray area in differentiating a sensor network from a mobile ad hoc network (MANET). A MANET does not rely on a fixed infrastructure, and typically consists on a scalable number of nodes that rely on each other for communication.

Designs of MANETs vary widely, with typical deployment numbers ranging from tens to hundreds to thousands, but each design faces common challenges. How do we build a network infrastructure that spans the breadth of the network in the most efficient method possible? How do we build a security mechanism suitable in a low-power, wireless environment? Sensor networks generally exacerbate these challenges because they typically consist of smaller nodes, with even less power, yet over a larger scale. An oft-cited example of a security system for sensor networks is U.C. Berkeley's SPINS [16], specifically designed for a SmartDust network [17]. This is a security protocol envisioned for the tiniest of devices, only millimeters in width, commonly referred to as microelectromechanical systems or MEMS. Although we continue to realize rapid improvements in semiconductors and chip fabrication at a rate close to Moore's Law, improved battery power and battery lifespan lag behind. Thus the challenges facing the Berkeley team, and others interested in the field, remain unique and daunting. We can see, however, that sensor nets are a subset of MANETs, even if all sensor nodes are not necessarily mobile. Earlier research focused primarily on securing MANETs [22, 11, 9]. Typically, sensor nets can make no assumptions about the computing power of the nodes or network infrastructure. On the other hand, there exists an implicit assumption in MANETs that power is less critical and computing power is more robust. Many of the challenges are common to both arenas, but it seems that research has forked; one path has developed novel approaches toward securing MANETs [7, 21, 12], and another set is finely focused on sensor net security [18, 16, 19]. The sensor net security contributions reveal the inherent deficiencies of a solution tailored for MANET when applied to sensor networking; often, the assumptions regarding power, and the bulky size of network communication is simply too massive. In this paper, the security considerations and proposed solutions specifically relate to sensor networks. However, strategies for securing MANETs and sensor networks often overlap and occasionally we may use these terms interchangeably while noting critical differences as necessary.

Regardless of the scenario into which the sensors are deployed, in most cases the networks not only monitor but also react decisively and intelligently to observed data, in essence they are intended to form smart spaces. The actual usages are as varied as physical phenomena, and while we have yet to realize the widespread deployment of sensor nets, these systems will provide foundations of realms in which observations are made and tasks are completed with little to no user intervention. This point alone reinforces the need for a novel approach to secure communication. Physical security is hypothetical, at best. In fact, most security designs assume physical compromise of one or many nodes, and sometimes describe the number of nodes that must be captured in order to render a security system useless. In certain foreseeable applications, the sensor network serves as a security system. In such a network, there are no wires, no firewalls, no locked server rooms — all countermeasures common in traditional network security. These fragile, inherently vulnerable systems generate traffic that is subject to detection, analysis, and interruption; these conditions are the consequences of incomplete security objectives, covered next.

1.1 Challenges of Secure Sensor Networks

Zhou and Haas contributed one of the earliest papers addressing the specific challenges of securing ad hoc networks [22]. Although the authors casually mention sensor networks, they make no claims regarding the viability of their solution for any arbitrary sensor net. In fact, this seems to be the tendency of most authors in the field, a clear indication of the diversity of purposes and associated variability of sensor nets. The primary issues of sensor network security are as follows:

- *Availability* relates to the survivability and uptime of the network. Availability can be attacked in numerous ways. Disabling power sources, jamming signals

(a denial of service attack), or confusing the network by disrupting the routing protocol — perhaps by compromising just a single node — are a few examples.

- By employing methods to guarantee *authenticity*, nodes can be sure that data in the system has originated from a peer node. Proper authentication schemes attempt to prohibit adversaries from masquerading as members of the system, and form the basis for authorization. Problems with authenticity are particular acute in sensor networks because of the wireless medium. Adversaries can easily eavesdrop or even inject messages into the network without physical barriers.

- *Integrity* ensures that data arrives at its destination exactly as it originated, i.e., the data has not been modified as it was transmitted, in-flight, or received.

- *Confidentiality*, achieved through encryption and cryptography, shields eavesdroppers from listening or deriving useful information from data. The importance of confidentiality varies according to the sensitivity of the application data and the purpose of the network. A sensor network transmitting troop locations would clearly require tighter confidentiality controls than one observing migratory birds. Key distribution, the primary emphasis of this paper, is a great example of a highly sensitive application that is common among most sensor networks.

- *Non-repudiation* ensures that messages are clearly associated with the message originator. By enforcing non-repudiation, a sensor net could possess stronger self-diagnostic attributes, i.e., nodes can use the non-repudiation property to determine if a peer node has been compromised.

Sensor network security must address the issues listed above, and each of those issues revolve around the central tenet of trust. Securing a sensor network involves developing a trusted computing base, a framework in which the constituents can readily identify the origin, destination, and purpose of the communication. Typically,

this trust is built by securely distributing keys, a topic covered later in this paper. By building trust within the system, the network is hardened against many types of attacks [9]. Some attacks are merely *passive*, such as eavesdropping, while others are *active* such as any attempts to replicate, modify, or delete data. Attacks could be *external* to the system, such as an adversary attempting to jam the signal or initiate a man-in-the-middle attack, or *internal*, perhaps a more delicate and dangerous issue and clearly linked to the concept of trust. Adversaries can launch internal attacks by compromising nodes via tampering, injecting malicious code, or other methods; these nodes may behave erratically and out-of-protocol, and a security system must not only be able to identify a compromised node but also safeguard the survival of the overall network. In fact, it is unreasonable to assume that nodes *will not* be compromised, especially given the lack of physical security, the fragility and wide distribution of the nodes, and the wireless access medium [11]. In most deployments, a single compromised node, or a small number of compromised nodes proportional to the population of the system is a trivial matter. On the other hand, denying known compromised nodes access and services to the still functional parts of the system is more difficult and clearly a pivotal issue. Ensuring the survivability and resiliency of the network — by protecting availability, confidentiality, authenticity, and so on — guides the development of sensor network security.

Chapter 2

Cryptography

Protecting the confidentiality of data is the primary function of cryptographic algorithms. By encrypting a plaintext message, transmitting the ciphertext, then decrypting ciphertext and recovering the message on the receiver, nodes can securely broadcast data on the network. By no means does encryption secure the communication against all attacks. For example, encryption could never prevent eavesdropping; but properly implemented cryptographic algorithms are well suited for sensor and mobile ad hoc networks. Since the design and purpose of MANETs vary, several existing cryptographic algorithms are worthy of consideration. However, we can assume that because of the limited computational resources, the need for *lightweight*, less computationally intensive algorithms is requisite in this domain.

2.1 Public Key Cryptography

Asymmetric (public/private key) cryptographic algorithms such as RSA encryption are too computationally intensive for most motes, and possibly too slow, but remain attractive in MANETs with more resources or renewable power [22, 11]. However,

the wholesale dismissal of this cryptographic category would be unwise especially considering the rate at which processing power increases, even in tiny sensor motes. In asymmetric cryptography, each user receives both a public and a private key. Using these keys for encryption and decryption serves a variety of purposes. For instance, our canonical secure-communication friends, Alice and Bob, can use asymmetric cryptography to secure a channel. The public key can be disclosed widely and still not pose a threat to the security. However, revealing the private key would constitute a grave breach of security. So how do Alice and Bob go about secure communication using asymmetric cryptography? First, they exchange public keys in plaintext. Next, Alice encrypts a message to Bob using Bob's public key. Since encryption and decryption are mathematical functions that are inverses of one another, only Bob's private key can be used to decrypt the message. So Alice sends the ciphertext to Bob, he uses his private key to decrypt, and reads the decrypted plaintext. However, this scheme alone does not solve the problem of *non-repudiation*. How can Bob be certain that Alice was in fact the original sender of the message? After all, anyone may have had access to Bob's public key. Fortunately, asymmetric cryptography includes a *digital signature* capability. If Alice had first used her own private key to encrypt the message and then used Bob's public key to encrypt the message for Bob's eyes only, Bob would have used his own private key and then Alice's public key to decrypt. If the result was a readable message, then certainly the message must have come from Alice, i.e., the message had been digitally signed by Alice.

Since the scheme does not rely on a single secret key, should Alice inadvertently reveal her private key to an adversary, only her key should be revoked. She could generate a new public/private key pair, re-distribute the public key, and immediately create a new channel for secure communication. But it is obvious that public/private key pairs for all users, or for all nodes in a sensor network, presents significant challenges in regard to both storage and communication cost.

2.2 Secret Key Cryptography

Thus, symmetric (secret or single key) algorithms emerge as a more attractive and viable strategy for arbitrary sensor nets. Secret key cryptography is ancient technology. It merely requires that users share some common secret and then use that secret to protect information. Block ciphers like RC5 or RC6 (for a description of RC5 and RC6, see [20]) are more suitable cryptographic algorithms because they require fewer computational resources [16, 18, 5]. However, key distribution becomes a more acute problem in symmetric systems, since such a system requires that the key be securely exchanged or deployed in an environment free from eavesdroppers, e.g. pre-deployment. Each solution to this problem comes with its own associated set of drawbacks, underlining the theme that security, especially in sensor networks, is a compromise. One novel suggestion notes that keys can be synchronously generated with precision on neighboring nodes [18, 16]; however a scheme that relies on synchrony, though, is likely vulnerable to attacks that affect the clock, skewing the keys and thus disabling communication [22]. Clearly, selecting the proper cryptographic mechanisms is an impressive challenge and no selection comes without drawbacks.

It is worth noting that neither an asymmetric or symmetric system is unbreakable or completely secure. An adversary with sufficient time, processing power, and enough ciphertext can break the encryption.

2.3 Hash Functions

Hash functions are examples of *one-way functions*. This means that it is impractical to derive the input from the output, in essence one-way functions are simple to calculate and difficult to invert. There are two simple characteristics of good hash functions. First, the output should be sufficiently random in that one should not

be able to identify likely input candidates from the output. A trivial hash function that takes "zebra" as input and returns the backwards spelling "arbez" as output quickly fails this condition. Second, a good hash function reduces the likelihood of collisions, meaning that no two unique inputs map to the same output. A function that maps both "zebra" and "elephant" to "Xe56zzf" is just as dismal a failure as the first example despite satisfying the first condition. A real example of a good hash function is the MD5 algorithm (see RFC 1321). MD5 is often used in sensor net security because of its utility in ensuring integrity, non-repudiation, and its utility in the creation of digital signatures. Remember that one could achieve non-repudiation using asymmetric cryptography. However, one can achieve the same effect or perform an integrity check with a hash function (also known as a *message digest*) without expending the same computational power which is a clear advantage in the context of sensor networks. For instance, consider the transmission of a long message. Encrypting the entire message is computationally intensive using traditional cryptography. However, a user can create a digital signature by calculating the hash of that message. Next, the user encrypts the hash output and sends it alongside the original message. At the other end, the recipient reads the message and calculates her own hash value. If her result equals the decrypted hash value from the sender, verification of the message integrity is complete. The net effect is a savings in processing power and communication cost.

2.4 Variable Encryption

The authors of [18] suggest employing various levels of encryption dependent on the sensitivity of the plain data. The strongest level of encryption should be applied to messages containing mobile code, i.e., data that could alter the behavior of the network. The second level of encryption is reserved for messages containing locations

of nodes, especially important in tactical battlefield type scenarios. A still weaker level protects application data, which when decrypted might still be meaningless to an untrained adversary, or perhaps reveals no information critical to network survival or operation. Weak does not imply unencrypted; even the weakest level makes use of a secret key and an application of the MD5 hash function to encrypt. This type of strategy could adapt well to the key distribution problem of symmetric cryptography by using a public/private key system to exchange secret keys. Because higher, more computationally intensive mechanisms are reserved for more sensitive data that is transmitted less frequently, an ad hoc network security solution can inch closer to the delicate balance between inadequate encryption and heavy computational burdens.

Chapter 3

Securing Sensor Networks

Securing sensor networks is a difficult problem because the design of the system must not only thwart common network vulnerabilities, but also defend against some weaknesses germane sensor networks. First, wireless communication introduces vulnerabilities not present in wired scenarios. Most of these result from the simplicity of eavesdropping; with air serving as the communication conduit, a listener need only a device capable of receiving transmissions at particular frequencies and operating in *promiscuous mode* to capture airborne network data. Similarly, an adversary could actively interfere with transmission signals at a known frequency. Researchers continue to develop strategies to evade, inhibit, or at least thwart eavesdroppers, but wireless network security is beyond the scope of this paper. We must assume that our communication channels are freely accessible (at a minimum, the traffic can be captured) to any adversaries.

Security is additionally challenging because there exists a high premium on energy in a sensor net deployment. Encryption and decryption are typically expensive operations in that they generally require significant amounts of encryption. Any resources consumed toward the end goal of network security are consequently re-

sources unavailable to use for the actual purpose underlying the initial deployment of the sensor network. In a sense, designers of sensor network security must develop solutions to defend against today's network compromise methods with yesterday's technology and computational power. A designer must assume that the adversary possesses an abundance of computing resources. Completing this scenario with a field full of comparably weak sensor nodes with limited lifespans clearly depicts the uphill battle.

Another unique characteristic of sensor networks is the tendency for deployments to be self-organizing. Self-organization implies that nodes are not configured prior to deployment. Only after the nodes have been fielded do they discover neighboring nodes, establish communication routes, share keys for encryption, and so on.

3.1 Vulnerabilities of Sensor Networks

Although self-organization is not a requirement in sensor network deployments, it is such a common property that we will discount the other alternatives in this paper. Self-organization opens up a new vulnerability for several reasons. Since it implies that nodes have a relatively high degree of autonomy and are generally surrounded by equivalent nodes, a compromise or capture of any node could pose a bigger threat in this environment compared to others. For contrast, consider a service like DHCP, the Dynamic Host Configuration Protocol. After configuring a server to run DHCP, previously unknown nodes can make a request to obtain a variety of network information including and IP address to use on the network. Since the network configuration is stored in the server running DHCP, the compromise of a member node does not necessarily represent a vast breach of security. An adversary would have to gain access to the DHCP server and alter the configuration. Since each sensor node is potentially one "hop" among many for data intended for distant nodes that may

lie miles away, such compromise poses far more sever consequences to the network's operation as whole because an attacker might be able to alter the network configuration, inject malicious instructions, or come up with new and creative ways to disable the network at will.

3.2 Attacks on Sensor Networks

Attacks on sensor networks can be categorized generally as either *passive* or *active*. In a passive attack, we assume that the adversary has the ability to eavesdrop on network traffic and possibly use the captured information (either plaintext or decrypted ciphertext) to do harm. An active attack implies that the adversary has captured one or more sensor nodes and can use the memory contents (e.g., recovered key rings) to compromise the network. Given the range of attacks relevant to sensor networks, some of which are described below, it is important to note that sensor networks must be inherently untrusted, and that confidentiality and authenticity remain high priorities in any security system.

Sleep Deprivation. Since sensor nodes typically operate on battery power and thus possess relatively limited operable lifetimes, energy conservation is at a premium. Communication protocols for sensor networks are designed to preserve this lifetime. One common strategy is to allow certain nodes to power-down, or sleep, on a rotating basis. Although this may introduce higher communication costs since more hops may be required to properly route traffic for source to destination, it has the added benefit or preserving each node's lifetime and in turn preserving the overall coverage and capability of the sensor network. Thus, a *sleep-deprivation attack* seeks to interrupt the normal sleep cycle, often by flooding the network with meaningless messages, or artificially inflating the connectedness of one particular node (i.e., if a

node determined that it was centrally useful to the network, it would likely forfeit any opportunity that it may have to sleep). Some of the attacks described below may also cause sleep deprivation as a side effect.

Spoofing, deception, selective forwarding. Several of the attacks common in sensor networks require the injection of an adversary that is more powerful than the sensor nodes themselves. This capability introduces a wide range of options for the attacker, such as the ability to communicate over longer distances, remain powered on indefinitely, and more advanced computational tasks. However, there are some general security vulnerabilities that are exposed simply by compromising one or more of the sensor nodes. After a sensor node has been compromised an adversary typically has a few basic tools. For one, a compromised node can *selectively forward* traffic. This might mean that a node stops forwarding traffic altogether although this might raise suspicion from other uncompromised nodes. A more subtle method would be to forward a few packets to limit the exposure of its malicious activity. Additionally we can generally assume that a compromised node can lie, cheat, eavesdrop on other conversations within range, modify or delete data that it receives, and otherwise manipulate the data with which it comes in contact.

Sinkhole attack. A sinkhole attack, first introduced in [8], attempts to modify the routing tables of the network such that all traffic flows through a (or at least one of several) compromised node. In this way, the compromised node can examine, modify, or redistribute traffic as the adversary sees fit. To execute a sinkhole attack, a compromised node must make itself appear as the most suitable next hop for a wide array of nodes. What makes a node more or less attractive depends on the routing protocol and the important attributes in each protocol. For example, nodes in a network designed to provide an immediate or brisk response would likely prioritize low latency links to minimize delay. A compromised node could thus

advertise or spoof the speed of its links. Another example is a protocol that credits high connectivity (i.e., a greater number of neighbors) and thus a compromised node would advertise its popularity among neighboring nodes regardless of whether or not a link was present.

Sensor networks that ultimately route all traffic to just a single base station are particularly vulnerable to sinkhole attacks because just a single compromise, followed by a successful sinkhole attack, can exert an enormous amount of influence over the complete function of the network.

Sybil. In a Sybil attack, a single node masquerades with multiple identities [2]. A Sybil attack is not unique to sensor networks; peer-to-peer networks were the initial area of concern. However, sensor networks featuring geographic routing protocols are particularly susceptible to Sybil attacks because a single node can seemingly exist in multiple places simultaneously. Thus, the Sybil attack seriously undermines fault-tolerance by wreaking havoc on distributed services. Routes that are assumed to be disjoint by members of the network may actually all transmit through a node or nodes compromised by the Sybil attack. Douceur demonstrates in his paper that the Sybil attack is virtually impossible to prevent in a network lacking a central authority without enforcing some unrealistic policies and parameters for communication.

Wormhole. A wormhole attack opens a low-latency (i.e., lower latency than normal multihop routing) link between two (generally colluding) nodes, routing traffic intended for one part of a network and transmitting it to another [6]. Unlike many of the other attacks described earlier, a successful wormhole attack often requires additional computing resources. This is because the creation of low latency link to distant nodes is rarely possible with the hardware present in sensor networks. However, injecting a laptop-class attacker (or several) into the network with more

powerful transmission capabilities than peer nodes is one such method of creating a wormhole. The malicious nodes communicate with the sensor nodes using the communication protocols mundane to the network, but communicate with one another in a different way, thereby disguising themselves from detection by the remaining trusted members of the sensor network. Two colluding nodes on either end of the wormhole link would likely understate their distances to other nodes, latencies, and so on to appear as attractive routing points and collect massive amounts of traffic. In this way, wormhole nodes often become sinkholes.

HELLO flood. Another kind of attack, the HELLO flood, is specific to sensor networks. First mentioned in [8], the attack exploits a feature common among many sensor network protocols. At startup, many protocols initiate a neighbor discovery phase during which nodes power on radios and say "HELLO" to others within range. Upon receipt of a "HELLO" message, a node generally assumes that some other node exists within normal radio range. After just a few messages have been exchanged, most nodes have a complete picture of their immediate vicinity and a routing topology logically forms in a self-organizing fashion. However, if an attacker featuring a more powerful radio was injected into the network and communicated "HELLO," not only would the attacking node's voice reach far and wide but that attacker might also be advertising attractive routing pathways through itself. Later, after convincing portions of the network that it is truly the best routing option, it might choose to ignore incoming messages, effectively disabling large portions of or even the entire network. In this fashion, the HELLO flood attack is a type of denial-of-service attack. Unlike some of the other kinds of attacks described in this paper, the HELLO flood attack does not require that an attacking node to create legitimate traffic to be successful. For instance, if a node first captured legitimate "HELLO" messages as they breezed through the air and then forwarded them on with a more powerful antenna, those messages would reach other nodes well beyond the actual

reach of the real sensor node's hardware. It's easy to see that this forwarding and redistribution leads to false network topologies and bogus routing information. An additional inherent weakness of protocols featuring "HELLO" messages, or really neighbor discovery in general, is that a node's identity and characteristics are revealed and exchanged during this step [7]. Since we seek to maximize efficiency and minimize the cost of information exchange to prolong sensor lifetime, it is advantageous for nodes to inform its peers of its critical characteristics, like amount of battery power remaining, connectivity, address, and so on. While any one piece of information is unlikely to result in widespread damage to the network, it is easy to see how the cumulative effect of information gathering can empower an adversary to manipulate the functions of the network.

Chapter 4

Key Management

According to Menezes et al., key management [13]:

- initializes system users within a (secure) domain;
- generates, distributes, and installs keying material;
- controls the use of keying material;
- updates, revokes, and destroys keying material;
- stores, backups or recovers, and archives keying material.

Any key management system must deal with the threat of compromise. Any attack that alters the confidentiality or the authenticity of the key or keys critically undermines the scheme.

All key distribution schemes lie somewhere in the spectrum between completely centralized authority and fully distributed authority.

4.1 Secure Routing

Secure routing is a primary example of an application in sensor networks that stresses the need for an efficient, yet fully secure (i.e. establishing confidentiality, authenticity, etc.) key management scheme. Routing information in a sensor network can be especially valuable since an adversary possessing such knowledge can wield powerful attacks on the newly-vulnerable network. Attacks such as the sinkhole or wormhole attacks discussed earlier rely on the fact that the routing information in a sensor network is insecure or can be recovered by an attacker. By obtaining the routing information, an adversary would discover the critical communication channels in the network, the location of well-connected nodes, and so on. This could greatly simplify the amount of work necessary to execute a successful attack.

4.2 Centralized Authority

To be fair, a centralized authority does feature a few advantages. For instance, centralization reduces the administrative burden significantly. Also, security personnel can focus their efforts on physically protecting a just a single or a few machines (the authorities) versus all connected nodes in the network. However, central authority introduces many drawbacks, especially in a sensor network. It creates single points of failure. Second, it requires fast connectivity. Requiring two neighboring nodes to establish authority and authentication with a central server prior to communicating with one another severely increases the communication costs. Since a central authority would likely have to manage multiple concurrent authentication requests, the authoritative machines would require more processing resources, bandwidth, and infinite lifespans. Simply put, these conditions are not likely found in a sensor network, and thus we can readily dismiss schemes reliant upon central authority.

4.3 Peer-to-Peer Authority

If centralized authority is not a viable option in sensor networks, what other options exist? One option is peer-to-peer (P2P) authority. Peer-to-peer technology decouples the traditional notion of servers and clients and distributes authority or any other service to the entity nodes themselves. In a sense, some sensor networks could be considered examples of P2P technology. Hubaux et al. introduce a P2P strategy [7] for sensor network security similar to PGP [23]. As in PGP, public-key certificates are issued by users, or in this case, individual sensor nodes. The primary difference between this strategy and PGP, however, is that this scheme does not rely on central directories for certificate distribution. In this fashion, this strategy allows already-certified nodes to certify other nodes, building virtual chains of self-organized trust. Such a mechanism solves many of the problems inherent to centralized authority, but potentially imposes higher communication costs and thus poor performance. Although peer-to-peer strategies scale well, and could suffice for many MANET deployments, the potential pitfalls render it ill-suited in a general sensor network environment.

4.4 Trusted Third Parties

A Trusted Third Party (TTP) system relies on the use of some entity other than the primary communicators to serve as an authenticator. The Kerberos network authentication service is a popular, well-known method of providing third-party authentication [10]. Kerberos is able to fulfill this role by supplying "credentials" in response to nodes' authentication requests. If Alice wanted to talk to Bob, she would make her intentions known (that she wants to communicate with Bob) to the server. The server would supply her credentials, along with a shared session key for Alice

and Bob to use during their conversation, to Bob. Upon receipt of Alice's credentials, Bob would know that the the other party is in fact Alice.

In this fashion, Kerberos, and other TTPs, can provide authentication services without the two end nodes knowing one another's identity.

Establishing trust between two nodes in a sensor network can prove complicated, so perhaps deferring the responsibility of authentication to a trusted third party would prove more efficient. If node b receives a message from node a that has been signed by a TTP verifying that indeed, node a sent the message, then node b need not spend any energy authenticating the message from node a since trust is implicit via the TTP. The primary challenge in sensor networks with Kerberos, like most third-party services, is that it re-introduces the problem of centralization. How can the locations of third-party authentication services be determined a priori in typical sensor networks that are typically designed for rapid deployment and routing topologies determined by self-organization? The only feasible was to make use of third-party authentication is if the third-parties are peer member nodes themselves.

4.4.1 Rich Uncles

Rich uncle is a term used to describe a node on the sensor network with more processing power and more substantial or perhaps infinite power resources. it is easy to see why rich uncles could be extremely useful in a sensor network filled with low power nodes. more power allows rich uncles to bear the major portion of the computational, communication, and security burdens while prolonging the lifetime of individual nodes and the overall network. these advantages allow rich uncles, among other purposes, to serve as trusted third parties. In a sensor network, the presence of rich uncles is the exception, not the rule. Described next is SPINS, a security system featuring rich uncles as trusted third parties [16].

4.4.2 SPINS

SPINS, short for *Security Protocols for Sensor Networks*, was created at the University of California. The powerful base stations featured in the network design are not so different from the peer nodes, except that they possess sufficient battery power to last for the entire lifetime of the network while communicating a significantly larger number of messages. These base stations have special properties; it is assumed that they may also connect to an external network, perhaps to relay data collected from the sensors for post-processing. Because of the special properties of base stations, the compromise of a base station can easily render the network useless. However, SPINS was designed to limit the damage from the compromise of a single node (a non-base-station node) to that single node alone. Each node is given a master key which is shared with the base station. This master key is used to derive other keys which are actually used for secure communication. To provide authenticated message broadcast, SPINS features a protocol called $\mu TESLA$. This is a modification of the original TESLA protocol in which the designers consider the considerable limitations of sensor networks [14, 15]. TESLA features authenticating an initial packet with a digital signature. But such signatures are computationally expensive to create and thus are a feature omitted from μTESLA. Instead, μTESLA uses only symmetric encryption and decryption mechanisms. For authentication purposes, a base station computes a *Message Authentication Code* (MAC) on the packet with a secret key. At first, the receiver does not possess the key needed to decipher the MAC and verify that the packet came from the base station. But once the key is received, the node will authenticate any packets it has in the buffer. The key management scheme in SPINS is significantly different than others discussed in this paper. To reproduce the benefits of asymmetric cryptography without the high overhead costs, symmetric keys are regularly broadcast on a delayed schedule instead. How is this achieved? To start, SPINS requires that nodes and base stations must be at least loosely time-

synchronized. This means that the system clock on each node must all be close in value and not out-of-bounds of some preset threshold value. This allows nodes to remember which keys have already been disclosed.

4.5 Distributing Key Management

Defining suitable strategies for secure routing and key distribution, along with decentralization, are prerequisites toward the ultimate goal of ensuring that a compromised node does not compromise the entire network. If the security system relies on symmetric cryptography, then the secret key must be agreed upon before deployment (i.e., it could be manually configured) or systematically generated on demand, based on a pseudo-random number generator or a mutually agreed upon seed. It is also evident that secret keys must be unique among any two nodes or small groups of nodes, or else the network is vulnerable to a complete system breach if with the compromise of only a single secret key, and the system loses the advantage of localized authority. Asymmetric cryptography relies on the protection of private keys, which is a less difficult problem than say, storing hundreds or thousands of arbitrarily large public keys in a system with limited memory, or polling a central authority for public keys for messages originating from an unfamiliar sender. Either approach spells the need to *distribute authority* across the entire network [22, 16, 11].

The authors of [11] provide a clear explanation of what constitutes a distributed authority model in the context of ad hoc networks. Essentially, distributed authority means that a node can send a trusted (authenticated) message based upon the collective authority of only a handful of neighboring nodes. This satisfies the condition of localization mentioned earlier. This collective authority is able to issue certificates, or create a digital signature, protecting the integrity and authenticity of the message and also enforcing non-repudiation. Since obtaining a certificate can be an expensive

procedure, it remains valid for some system-defined duration after it has been issued (determining the valid/renewal period is yet another challenge). Note that such a certificate service would also possess a revocation scheme to immediately deny the rights of any suspected compromised node.

4.5.1 Threshold Secret Sharing

Imagine that a system of N nodes has been deployed. Let us assume that at the time of system deployment there exists a *system secret key* SK, which is divided into "shares," denoted as P_{vi} where the concatenation of all P_{vi} would equal SK. These shares are distributed initially to K (or a multiple of K) nodes at startup, thus forming the initial trusted base. The number of nodes that receive shares depends on the size of N and the spatial locality of nodes in net. It is important to distribute a set of shares into enough neighborhoods such these shares could be collected within a small number of hops (note that 1 would be optimal). Note that by distributing shares, no single node contains the SK; an adversary would have to break compromise at least K nodes to recover SK. When an untrusted node seeks to join the network of trust and begin transmitting messages, it seeks to obtain a certificate that must be signed by the system secret key SK. When such a request is made, a local coalition of K secret share holders forms dynamically. We denote share holders as $v_i, 1 \leq i \leq K$. Each v_i provides the requester with a *partial certificate* SK_i, derived from the partial share P_{vi} belonging to v_i. The requester must collect K partial certificates to obtain its complete certificate, now signed by SK. We can also assume that the requesting node, at this point, can generate its own P_{vi} and become an authenticating node (this is covered in detail in [11]; it is a secondary issue here).

In this threshold secret sharing system, K becomes the threshold value. Why

is this advantageous in a sensor network setting? First, by selecting K such that $1 < K < N$ where N is the number of nodes in the system, several problems are minimized simultaneously. First, by distributing a system secret key among a sufficient number of K nodes, an adversary would be required to compromise K share holders to recover SK. Selecting a value of K less than N improves scalability by lessening overhead and ensures that the system can still issue authenticated certificates since an adversary is required to compromise $(N - K + 1)$ share holders to terminate the services. Similar to other facets of ad hoc network security, threshold secret sharing still faces challenges. Could SK be vulnerable over time, given an adversary with sufficient computing resources (yes), and how can SK be periodically refreshed and re-distributed in the system without any centralized services? The authors acknowledge this as an open issue.

4.5.2 Probabilistic Key Management

One promising area of research for distributed key management is probabilistic key management. Essentially, a probabilistic approach describes a key management scheme where any two arbitrary nodes can establish secure communication with a certain probability. This approach trades a degree of connectivity for increased efficiency, lower storage overhead, and reduced communication totals. Obviously, the probability must be sufficiently high as a sensor network deployment often spans large areas and a high number of nodes. Probabilistic approaches rely on pre-deployment key assignments. The key sizes are intentionally small, and can be randomly distributed among the sensor nodes. With a random deployment, two nodes communicate and determine if they each have a key (or keys) in common. If this condition holds, the two nodes will establish a secure communication route; otherwise, the nodes will have to communicate through neighboring nodes to deliver messages. The advantages to this type of pairwise communication are many. Most notably, the

compromise of a single node or a set of nodes does not reveal the network's secrets. Further, there is no central key! Clearly, the chief disadvantage is that not all node-to-node communication routes can be secured, which implicitly builds in network latency, and perhaps shortens the overall lifespan of the network because nodes may generate more traffic to route messages.

4.5.3 Eschenauer and Gligor

There are clear disadvantages with using either of the most obvious key pre-distribution schemes. If only one or a few keys is used for secure communication, an adversary need only compromise as little as just a single node to reveal the network's secret. Pairwise key pre-distribution, in which each node is pre-configured with a different key for communication with $n-1$ peer nodes in the distributed sensor network, avoids such simple network compromise. However, this strategy comes with its own drawback of scaling exceptionally poorly. In this kind of scheme, each node is responsible for the storage of $n-1$ nodes, and requires that $n(n-1)/2$ keys be generated in a sensor network deployment. Pairwise distribution is a reasonable strategy for sensor networks, or nodes without stringent storage constraints or processing power, but such a strategy rapidly fails under deployments with a massive number of low-power nodes because the total amount of communication scales with n.

Eschenauer and Gligor were the first to propose a probabilistic strategy that remedies the pairwise pre-distribution scaling problem at the cost of connectivity [3]. Initially, a pool of P keys is generated and each member node in the network randomly selects k keys from P prior to deployment to complete its *key ring*. With each additional key on the key ring, a node increases the probability that it will share that key in common with a neighboring node and thus be able to communicate securely with the neighbor. This scheme also provides a process for revocation,

whereby certain keys can be revoked if nodes determine or even suspect compromise of a particular key. The authors point out that key ring sizes can remain small in most deployments without significant impact to the likelihood that any two nodes have a key in common. For example, when each node stores just 75 keys on its ring out of an initial pool of 10,000 keys, any two neighboring nodes will have a key in common with probability 0.5.

At initialization, nodes begin the *shared-key discovery* phase to establish the network topology and determine if they share keys with neighbors within communication range. Although this phase sends messages in plaintext, the nodes use unique identifiers to describe keys in their possession and do not reveal any information considered fundamentally critical to the security of the network. Two nodes establish a link only if they share a key (or keys) in common.

The next step is *path-key establishment* during which two nodes without a common key but lie within communication range (connected by two or more links after shared-key discovery). A node can initiate this by selecting an unused key on its key-ring and sending that to the other neighbor via the secure links established during shared-key discovery. Upon receipt and decryption, the other node can add the new key to its ring and now communicate securely over the newly formed channel thus increasing the redundancy and robustness of the network.

Note that after these two phases, the resulting graph of the network topology is fully-connected. This property ensures that any two arbitrary nodes can communicate with each other via one or more hops.

When an individual node is compromised, it is imperative that the network revokes its entire key ring. When revocation is required, a controller node possessing a wider communication range and might be mobile broadcasts a signed message including k key identifiers to be revoked. This seems to be an obvious weakness

since the presence of a controlling, more powerful node is not guaranteed in a sensor network. Since uncompromised nodes must revoke keys that have potentially been compromised, existing links may no longer be valid. After revocation, performing shared-key discovery followed by path-key establishment reconfigures the network. Fortunately, since the number of revoked keys on the key ring is small compared to the large number of keys in the initial pool, it is highly likely that only a few nodes and links are affected by the entire process.

4.6 Pairwise Key Establishment

Naïve approaches to secure pairwise communication include assigning a pool of keys P to each node pre-deployment. Each key would be shared with only one other node. This approach requires each node to store $n-1$ keys, with $\frac{n(n-1)}{2}$ different keys in the group where n is the number of nodes. Since sensor nodes are typically ill-equipped to handle such massive storage requirements, especially for any considerable key size, such a basic approach would soon fail or leave no room remaining for computation and fulfilling the mission of the network. Pairwise communication is an important asset in a sensor network because it eliminates the chance that a compromise of one or just a few nodes will jeopardize the security of the network as a whole. But the fundamental challenge of keys, key distribution, and establishing pairwise communication among nodes is that nodes can not store large numbers of keys — as illustrated by the naïve approach described above — but without this large key pool, it is far more difficult to achieve full pairwise communication.

To improve upon Eschenaur and Gligor's strategy of random pre-deployment key distribution, Di Pietro et al. introduce a pseudo-random key deployment scheme [1]. Their work attempts to establish pair-wise communication with a certain probability. Their first approach, the *direct protocol*, establishes security with fixed probability.

The second *cooperative protocol*, allows the network designer or administrator, or perhaps code within the nodes themselves to adjust the security level based on current circumstances. Heightening the security is directly proportional to the overhead imposed on the processor.

The primary considerations in the design of the direct protocol are memory efficiency, energy efficiency, and resiliency against attack from corrupted sensors.

For each sensor a, a pseudo random-number generator generates the indices of the keys that a will store. The generator uses a publicly known seed dependent on a. Since the seed is known, the k indices of the keys assigned to a can be computed by any other peer node. This imposes some additional storage cost: each node must store not only keys but also the index of keys. The storage of indices proves to be fundamentally useful data during *authentication*. Because the indices are known, sensor b can prove its identity by proving that it knows the keys it is supposed to know. Without this information, a sensor identity could be fabricated because multiple copies of the same key are distributed in the network.

In most designs, the most significant cost of communication is the transmission of messages. Since this design greatly reduces the amount of messages that must be transmitted to neighboring nodes at initialization compared to other schemes, perhaps this trade-off is indeed worthwhile. In fact, this scheme reduces the number of required messages for key discovery to zero because nodes can already compute the keys another node should have. This yields another additional security feature. Nodes no longer must respond to incoming request from unknown devices because the initial message received is already encrypted by a session key. Instead, each node must run at most k executions of the pseudo-random number generator, and k lookups in memory where k is the number of keys assigned to each sensor.

We assume that two nodes a and b can establish a secure channel if a and b share

a key in common, an event denoted as E below. We compute the probability $Pr[E]$ as follows:

$$Pr[E] = 1 - Pr[\bar{E}] = 1 - \frac{\binom{P-k}{k}}{\binom{P}{k}} \tag{4.1}$$

The probability of channel existence varies only with k and P, and not with n, the size of the network. By using this approach, $Pr[E]$ is remarkably high even when k is a tiny fraction of P. Because the security property is not dependent on n, this design exhibits an exceptional scaling ability.

To communicate securely using the direct protocol, node a first computes all of the keys it has in common with node b. As mentioned earlier, the publicly known seed of the pseudo-random generator allows a to perform a simple computation to get the indices of the b's keys. The keys in common are then XOR'ed together to form a session key $K_{a,b}$. The destination sensor, b, performs a similar computation after receiving the first message from a. An adversary must know all the of the keys that were used to construct $K_{a,b}$ to corrupt the channel. This requirement introduces an interesting side effect. Intuitively, it seems that increasing k would increase the security of any channel. However, although it does increase the likelihood that a secure channel exists between any two nodes, it may actually lead to a decrease in channel security and strength of the key. Note that when $k = P$ the probability of channel corruption equals 1. This result makes sense because it means that any node can correctly compute any key used between any other two nodes. In a real-world scenario, imagine a network with a set w of corrupted nodes. As w increases, so does the likelihood that w as a coalition will know k keys. Thus, more session keys can be generated leading to an increased probability of channel compromise. More formally, a coalition of nodes $w \in W$ must know all of the keys $V_a \cup V_b$ (where V_i is the set of keys belonging to node i) to compromise the secure channel between a and b. The

Co-operative_Protocol(b :sensor)
Input: the receiving sensor;
Output: $k_{a,b}^{\mathcal{C}}$ and the set of co-operating sensors $\mathcal{C}$

1. Generate set $\mathcal{C}$;
2. Set time-out Δ;
3. $k_{a,b}^{\mathcal{C}} = 0$;
4. **for all** $c \in \mathcal{C}$ **do begin**
5. $\quad k_{a,c} = \text{Direct\_Protocol}(c)$;
6. $\quad a \rightarrow c :< a, c, E_{k_{a,c}}(\texttt{req\_key}||b) >$
7. **end**;
8. $\mathcal{C}' = \mathcal{C}$;
9. **while** ($\mathcal{C}' \neq \emptyset$ **and** (**not** elapsed(Δ))) **do begin**
10. $\quad a \leftarrow c :< c, a, E_{k_{c,a}}(HMAC(ID_a, k_{c,b})) >$;
11. $\quad s = E_{k_{a,c}}^{-1}(E_{k_{a,c}}(HMAC(ID_a, k_{c,b})))$;
12. $\quad \mathcal{C}' = \mathcal{C}' - \{c\}$;
13. $\quad k_{a,b}^{\mathcal{C}} = k_{a,b}^{\mathcal{C}} \oplus s$
14. **end**
15. $k_{a,b}^{\mathcal{C}} = k_{a,b} \oplus k_{a,b}^{\mathcal{C}}$;
16. $a \rightarrow b :< a, b, E_{k_{a,b}}(\mathcal{C})||\mathcal{H}(k_{a,b}^{\mathcal{C}}) >$;

Figure 4.1: The Cooperative Protocol, in Di Pietro, et al. [1]

coalition must also possess this same totality of keys to masquerade as either node, since a node must locally compute the session key depending on $V_a \cup V_b$.

While the direct protocol offers significant advantages to other schemes because it shifts the energy burden from transmission to computation (for a net gain in energy), it comes with drawbacks. The preceding paragraph illustrates one limitation. How does the network react if a breach has been detected? How might the security be increased temporarily if a crisis arrives? The cooperative protocol (described in figure 4.1) does introduce more communication among nodes, and thus it consumes more

energy. To establish a secure channel between two nodes a and b, sensor a chooses a set $C = c_1, \ldots, c_m$ of cooperative sensors such that $a, b \notin C$ and $m >= 0$. Node a then transmits a request for cooperation to every node $c \in C$. This encrypted request (with $k_{a,c}$) includes an ID for node b. Each cooperating sensor c transforms the original session key with $b, k_{c,b}$ by first building it according to the direct protocol and then creating a *share* by hashing $k_{c,b}$ with id of a.

Node c then transmits this share back to a encrypted with their original session key $k_{a,c}$. After a has received shares from every node $c \in C$, a computes $k_{a,b}$ and combines it with the collected shares to create a new cooperative channel $k^C_{a,b}$. Because of the hash function, this key is one-way and non-invertible under any circumstances, thus no information transmitted over a channel encrypted with this key can be discovered by an adversary.

Sensor a then transmits to sensor b using key $k_{a,b}$ a list of all $c \in C$ along with the newly computed $H(k^C_{a,b})$. Since b now has the list of the nodes in C, b can compute $k^C_{a,b}$ and thus complete the formation of the secure channel.

Chapter 5

Determining the Set C

The authors of [1] do not specify the method by which set C is chosen while executing the cooperative protocol. However, the basis for selection is likely to factor heavily into communication cost and effectiveness of the network and thus it is worthwhile component on the cooperative protocol to examine. It is important to note that establishing a secure channel with the cooperative protocol need only occur once, just prior to sending the first message over the encrypted channel. The cooperative protocol requires that sensor a transmits $m + 1$ messages ($m = |C|$), and receives m replies. Thus the cost for the network as a whole would be $2m + 1$ potentially multi-hop messages per new secure channel. Although this represents a significant cost increase over the direct protocol, the result is a drastically more secure channel.

The authors suggest three basic schemes to the reader:

- *Neighbors* insists that C is drawn from nearby nodes. By limiting the average hop length, one would expect to see a complementary reduction in the number of broadcasts and hop length per message. However, this strategy's primary weakness is its susceptibility to localized attacks. What would a node do if all of its immediate neighbors had already been corrupted? The network designer

might consider supplementing the set selection strategy to guard against local failures.

- *Random* selection will almost always significantly increase the overhead costs. Imagine if a sensor node chose a peer among 100,000 that sat 100 hops away. Imagine choosing two at that distance. We would expect the additional incurred cost to establish such secure channels to be dramatically increased. Selecting nodes at random for inclusion in C does effectively reduce the threat posed by a localized attack.

- *Favorites* influences the composition of set C by suggesting (or requiring) the selection of nodes with special properties, such as tamper-resistance or nodes that are rich uncles. The set C would still be drawn randomly, however, the selection pool would be limited to just nodes with the special properties. Unless every node (or nearly every node) was given this special property, We would expect to see fewer message transmissions than in the random strategy (especially if these nodes were placed strategically), however, "playing favorites" will likely introduce its own disadvantages. Additionally, any multi-hop messages occurring while operating in favorites mode are a better use of energy since the key shares have been derived from nodes with special properties and thus are less susceptible to compromise. But the drawbacks certainly include the increased in demand for services which could lead to network hotspots, congestion, and latency. These hotspots might also provide hints to adversaries who might be trying to prioritize targets.

5.1 Experimental Design

Testing various conditions for the selection of C requires a suitable test environment, one that could simulate a sensor network deployment. An original *network simulator*

(hereafter known simply as the "sim") was constructed to allow the tester full control over the variables specified in [1].

5.1.1 Network Topology

In the sim, each node is modeled as though it were a part of a real-world deployment. As the nodes are created, the sim generates topologies by dispersing n nodes randomly across a virtual, square plot. Then, neighboring nodes are determined according to radio range. This gives rise to a network topology in the form of an undirected connected graph.[1] The sim operator is able to specify the minimum and maximum values of neighbors to constrain the degree of each vertex (or node) in the graph (or network). For these purposes, we have constrained only the minimum degree to ensure that the neighbors condition never fails.

5.1.2 Routing

Such an experiment requires only a very simple routing implementation. Each node performs only two atomic communication operations: broadcast, during which a message is transmitted to all nodes within radio range, and receive. While the designers of any real-world deployment would undoubtedly carefully consider routing options, equipping each of the nodes in this model with a simple *shortest-paths* routing model is sufficient because we have the luxury of an error-free environment. Neither does the simulator require a transport protocol. In fact, introducing various network or transport protocols could spuriously affect the accuracy of the results.

[1]Undirected indicates that messages can flow in either direction. In a connected graph, it is possible to visit all vertices along existing edges.

5.1.3 Protocols

Each node is able to generate and process messages according to the direct and cooperative protocols described in [1]. Since the additional energy required to perform encryption, calculate hash values, and so on scales linearly with the number of messages created, we choose to omit those functions for simplicity.

5.1.4 Test Design and Configuration

We begin by creating network topologies while varying n, the number of nodes. Since we are not modeling attacks on the network or the set of compromised sensors w as originally described in [1], we assign k equal to p, such that all nodes possess all keys. In effect, this renders the values of k and p irrelevant because it ensures that every node can establish a secure channel using the direct protocol with any other node in the network. Similarly, it is unimportant to vary m, the size of C, during this experiment. While increasing m decreases the channel corruption probability in the original experiment, an increase here would only linearly increase the number of messages per connection. The primary determinant for the results of this experiment will be the path length for the transmitted messages.

For each topology we select at random $n \cdot 10$ pairwise connections to establish during each run of the simulator. This number generates enough traffic to provide consistent results. It is important to the integrity of the experiment to establish the same connections for each selection strategy. In other words, the geographical distribution and the order of connection establishment remains fixed for each run as we iterate through the three selection strategies: one-hop neighbors, random, and favorites.

Since each connection from $a \rightarrow b$ relies on m nodes in C, and $a, b \notin C$, each node

must have a minimum degree of $m+2$ to ensure that it has enough one-hop neighbors to make a complete set C. Similarly, at a minimum, $m + 2$ nodes are assigned properties like tamper-resistance to be included in the set of "favorites." The authors of [1] also fail to specify in the cooperative protocol if the direct protocol established in step 5 runs for each new connection. For example, if a had already established a direct connection with d because d had already been a part of a cooperating set in the establishment of a previous connection, would a need to reestablish the connection with d using the direct protocol? Generally, this practice seems wasteful, but it could be useful in the event that the threat model had changed and d had become compromised. Since our simulator does not model threats, we choose to cache established connections to avoid redundancy.

5.2 Results and Analysis

Two tests were performed to examine set selection policy. The number of nodes in the test varied between 40 and 100. Only $m+2$ nodes, the minimum acceptable value, were given tamper-resistant properties. For each test, $m = 4$, the minimum value used in the original experiment [1]. Assigning a larger value to m would only linearly increase the amount of traffic without altering the behavior because each channel requires $2m + 1$ transmitted messages. Each test (per value of n) was executed 10^2 times.

Figure 5.1 depicts the first test results. As the nodal degree increases, so does the number of broadcasts. However, the nearest-neighbors policy steadily outperforms both the random and criteria-based selection policies. The most-connected nodes in each topology exhibit an especially profound increase in communication when using either the random or criteria-based scheme.

Figure 5.2 depicts a similar result when $n = 100$. Similar to the performance

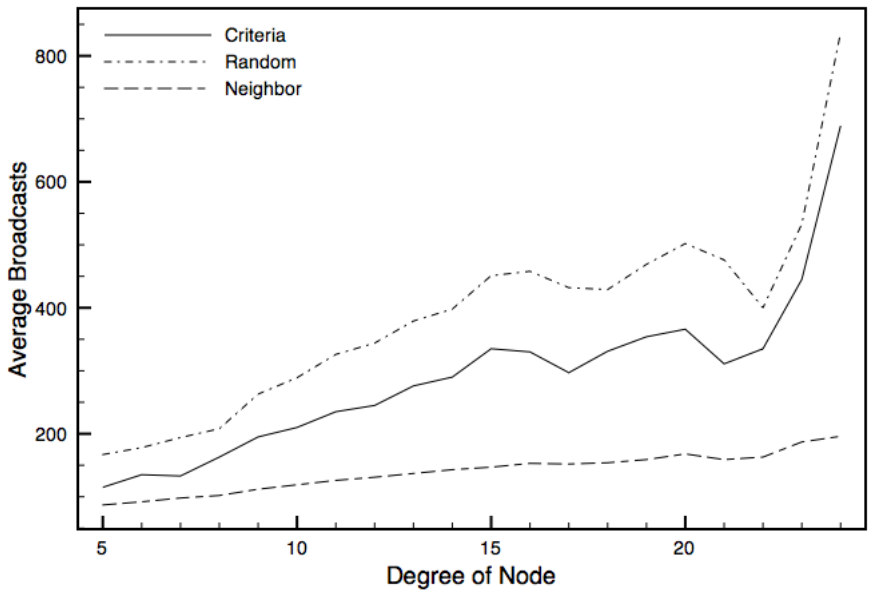

Figure 5.1: Average broadcasts per node, 40 nodes per test

in the previous test, nodes with degree of approximately 15 experience a four- to five-times increase in communication overhead when not using the nearest-neighbors policy. However, this graph indicates that there may be an upper limit to the total amount of additional cost a node must endure in another scheme. At first glance, this seems to be a highly counter-intuitive result. Figure 5.3 provides a partial explanation: there are simply few nodes with high degree, and with a data sample size of only 10^2 instances may not yield enough information to smooth out the curve. Furthermore, it logically follows that nodes with high degree must be located somewhere near the center of the topology, or at least among a relatively tight cluster of nodes. We could then infer that this hyper-connected node's neighbors also exhibit exceptionally high degree. These properties suggests that several alternative paths exist adjacently that do not include a particularly hyper-connected node. In other words, when determining the shortest-path between two endpoints, there are likely many paths of equal length to to choose from and thus the communication burden

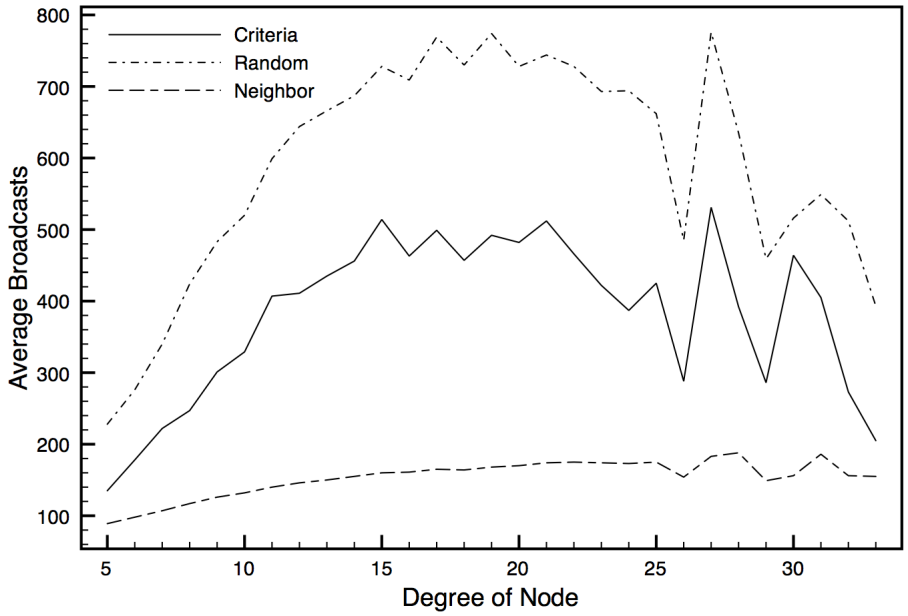

Figure 5.2: Average broadcasts per node, 100 nodes per test

is lessened on a per node basis.

5.3 Conclusions and Future Work

There exist several options worth exploring further. Introducing an energy model, one in which nodes cycle through sleeping and wakeful periods, would more closely resemble a real-world scenario. The simulation could also run over a simulated time period, during which nodes might actually expire after depleting energy resources. These additions would allow a user to inject attacks. Modeling attacks and introducing a set w of corrupted sensors would allow the measurement of the networks' fault tolerance. One interested in examining how this system functions over time would want to consider varying more conditions.

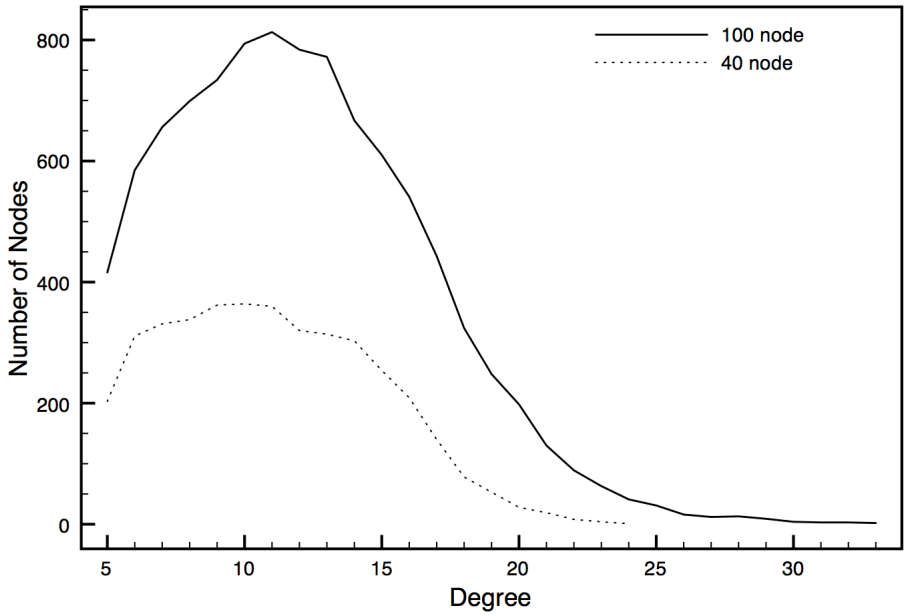

Figure 5.3: Total number of nodes with degree x

Varying the number of "favorites," rather than merely supplying each topology with $m + 2$ favorite nodes, would most likely alter the behavior of the favorites strategy such that it more closely resembled either random or neighbors. Coupling the neighbors strategy with the favorites strategy, i.e., to program the nodes such that each node was inclined to choose nearby nodes with special properties could have real-world application by minimizing the number of multi-hop messages while fortifying the reliability of channels.

The results indicate that using a scheme other than nearest neighbors would more rapidly deplete the scarce energy resources available in a sensor network deployment. However, to suggest that the nearest-neighbors approach is the only feasible solution would be foolish. This simulation modeled an ideal scenario: one free of attacks, enemies, and broken-down or compromised sensors. These were reasonable test conditions because establishing pairwise secure channels has only a one-time initial cost

and it is reasonable to expect the formation of many channels at system start up. This thesis demonstrates the considerable increase in communication cost for potentially more secure, or less error-prone options.

References

[1] Di Pietro, R., Mancini, L. V., and Mei, A. Random key-assignment for secure wireless sensor networks. In *First ACM Workshop Security of Ad Hoc and Sensor Networks (SASN '03)* (October 2003), pp. 62–71.

[2] Douceur, J. The sybil attack. In *Proceedings of the IPTPS02 Workshop* (Cambridge, MA, March 2002).

[3] Eschenauer, L., and Gligor, V. A key-management scheme for distributed sensor networks. In *ACM Conference on Computer and Communication Security* (Washington, D.C., November 2002), pp. 41–47.

[4] Estrin, D., Michener, W., Bonito, G., and the workshop participants, Eds. *Environmental Cyberinfrastrucure Needs For Distributed Sensor Networks: A Report from a National Science Foundation Sponsored Workshop* (La Jolla, CA, August 2003), Scripps Institute of Oceanography.

[5] Hu, Y., Perrig, A., and Johnson, D. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *The 8th ACM International Conference on Mobile Computing and Networking* (September 2002).

[6] Hu, Y., Perrig, A., and Johnson, D. Wormhole detection in wireless ad hoc networks. Tech. rep., Rice University Department of Computer Science, 2002.

[7] Hubaux, J., Buttyan, L., and Capkun, S. The quest for security in mobile ad hoc networks. In *Proceedings of the 2nd ACM Symposium on Mobile Ad Hoc Networks (MobiHOC)* (October 2001).

[8] Karlof, C., and Wagner, D. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols 1*, 2–3 (September 2003), 293–315.

[9] Kärpijoki, V. Security in ad hoc networks. In *Proceedings of the Helsinki University of Technology, Seminars on Network Security* (Helsinki, Finland, 2000).

[10] Kohl, J., and Neuman, B. The Kerberos network authentication service, 1991. Tech. report.

[11] Kong, J., Zerfos, P., Luo, H., and Lu, S. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *Ninth International Conference on Network Protocols (ICNP '01)* (2001), pp. 251–260.

[12] Luo, H., Zerfos, P., Kong, J., Lu, S., and Zhang, L. Self-securing ad hoc wireless networks. In *7th IEEE Symposium on Computers and Communications (ISCC '02)* (2002).

[13] Menezes, A., van Oorschot, P., and Vanstone, S. *Handbook of Applied Cryptography*. CRC Press, 1997.

[14] Perrig, A., Canetti, R., Song, D., and Tygar, J. Efficient and secure source authentication for multicast. In *Network and Distributed System Security Symposium (NDSS '01)* (February 2001).

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

[16] Perrig, A., Szewczyk, R., Wen, V., Cullar, D., and Tygar, J. SPINS: Security protocols for sensor networks. In *Proceedings of MOBICOM* (2001).

[17] Pister, K. S. J., Kahn, J. M., and Boser, B. E. Smart dust: Wirless networks of millimeter-scale sensor nodes, 1999.

[18] Slijepcevic, S., Potkonjak, M., Tsiatsis, V., Zimbeck, S., and Srivastava, M. B. On communication security in wireless ad-hoc sensor networks. In *11th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (2002), pp. 139–144.

[19] Undercoffer, J. L., Avancha, S., Joshi, A., and Pinkston, J. *Security for Sensor Networks*. Kluwer Academic Publishers, January 2004, ch. 12, pp. 253–275.

[20] Unknown, RSA Laboratories. What are RC5 and RC6? Web site, url: http://www.rsasecurity.com/rsalabs/faq/3-6-4.html.

[21] Yang, H., Zhong, G., and Lu, S. Network performance centric security design in manet. *Mobile Computing and Communications Review 1*, 2 (2002).

[22] Zhou, L., and Haas, Z. J. Securing ad hoc networks. *IEEE Network 13*, 6 (1999), 24–30.

[23] Zimmerman, P. *The Official PGP User's Guide*. MIT Press, Cambridge, MA, 1995.