# Distributed Detection of Node Replication Attacks in Sensor Networks

**Bryan Jeffrey Parno**

**2005**

Advisor: Prof. Perrig

# Distributed Detection of Node Replication Attacks in Sensor Networks

**Bryan Parno**

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania
May 2005

# Distributed Detection of Node Replication Attacks in Sensor Networks

Approved by:

[illegible]

Professor Adrian Perrig

[illegible] Apr 23, '05

Professor Dawn Song

Date Approved 23 April 2005

# Abstract

The low-cost, off-the-shelf hardware components in unshielded sensor-network nodes leave them vulnerable to compromise. With little effort, an adversary may capture nodes, analyze and replicate them, and surreptitiously insert these replicas at strategic locations within the network. Such attacks may have severe consequences; they may allow the adversary to corrupt network data or even disconnect significant parts of the network. Previous node replication detection schemes depend primarily on centralized mechanisms with single points of failure, or on neighborhood voting protocols that fail to detect distributed replications. To address these fundamental limitations, we propose two new algorithms based on emergent properties [17], i.e., properties that arise only through the collective action of multiple nodes. Randomized Multicast distributes node location information to randomly-selected witnesses, exploiting the birthday paradox to detect replicated nodes, while Line-Selected Multicast uses the topology of the network to detect replication. Both algorithms provide globally-aware, distributed node-replica detection, and Line-Selected Multicast displays particularly strong performance characteristics. We show that emergent algorithms represent a promising new approach to sensor network security; moreover, our results naturally extend to other classes of networks in which nodes can be captured, replicated and re-inserted by an adversary.

# Contents

**1 Introduction** **1**

**2 Background** **4**
- 2.1 Goals . . . 4
- 2.2 Sensor Network Environments . . . 4
- 2.3 Adversary Model . . . 5
- 2.4 Notation . . . 6

**3 Previous Protocols** **7**
- 3.1 Centralized Detection . . . 7
- 3.2 Local Detection . . . 8

**4 Preliminary Approaches** **9**
- 4.1 Node-To-Network Broadcasting . . . 9
- 4.2 Deterministic Multicast . . . 10

**5 Randomized Multicast** **12**
- 5.1 Assumptions . . . 12
- 5.2 Protocol Description . . . 13
- 5.3 Security Analysis . . . 14
- 5.4 Efficiency Analysis . . . 16

**6 Line-Selected Multicast** **18**
- 6.1 Overview . . . 18
- 6.2 Protocol Outline . . . 19
- 6.3 Analysis . . . 20

**7 Simulations** **22**

**8 Discussion** **24**
- 8.1 Protocol Comparison . . . 24
- 8.2 Synchronized Detection . . . 25
  - 8.2.1 High Noon . . . 25
  - 8.2.2 Time Slots . . . 26
  - 8.2.3 Security Requirements . . . 26
- 8.3 Authentication . . . 26
  - 8.3.1 Public Key Security Adjustments . . . 27
  - 8.3.2 Symmetric Alternatives . . . 27

8.4 Masked-Replication Attacks . . . . . 28
8.5 Emergent Properties . . . . . 29
8.6 Related Work . . . . . 30

**9 Future Work** **31**

**10 Conclusion** **32**

**A Network Topologies** **33**

# Chapter 1

# Introduction

The ease of deploying sensor networks contributes to their appeal. They can quickly scale to large configurations, since administrators can simply drop new sensors into the desired locations in the existing network. To join the network, new nodes require neither administrative intervention nor interaction with a base station; instead, they typically initiate simple neighbor discovery protocols [6, 13] by broadcasting their prestored credentials (e.g., their unique ID and/or the unique ID of their keys).

Unfortunately, sensor nodes typically employ low-cost commodity hardware components unprotected by the type of physical shielding that could preclude access to a sensor's memory, processing, sensing and communication components. Cost considerations make it impractical to use shielding that could detect pressure, voltage, and temperature changes [11, 33, 36] that an adversary might use to access a sensor's internal state. Deploying unshielded sensor nodes in hostile environments enables an adversary to capture, replicate, and insert duplicated nodes at chosen network locations with little effort. Thus, if the adversary compromises even a single node, she can replicate it indefinitely, spreading her influence throughout the network. If left undetected, node replication leaves any network vulnerable to a large class of insidious attacks. Using replicated nodes, the adversary can subvert data aggregation protocols by injecting false data or suppressing legitimate data. Further, blame for abnormal behavior can be spread across the replicas, reducing the likelihood that any one node exceeds the detection threshold. Even more insidiously, node replicas placed at judiciously chosen locations can revoke legitimate nodes and disconnect the network by triggering correct execution of node-revocation protocols that rely on voting schemes [6, 10, 13, 27].

Previous approaches for detecting node replication typically rely on centralized monitoring, since localized voting systems [6, 27] cannot detect distributed replication. Centralized schemes require all of the nodes in the network to transfer a list of their neighbors' claimed locations[1] to a central base station that can examine the lists for conflicting location claims. Like all centralized approaches, this creates a single-point of failure. If the adversary can compromise the base-station or interfere with its communications, then the centralized approach will fail. Also, the nodes surrounding the base station are subjected to an undue communication burden that may shorten the network's life expectancy.

In this paper, we use two different emergent algorithms to provide the first examples of globally-aware distributed node-replication detection systems. The emergent nature of our algorithms makes them extremely resilient to active attacks, and both protocols seek to minimize power consumption by limiting communication, while still operating within the extremely limited memory capacity of typical sensor nodes. An emergent algorithm leverages the features that no individual node can provide, but that emerge through the collaboration of many nodes. Our first protocol, Randomized Multicast, distributes location claims to a randomly selected set of witness nodes. The Birthday Paradox predicts that a collision will occur with high probability if the adversary attempts to replicate a node. Our second protocol, Line-Selected Multicast, exploits the routing topology of the network to select witnesses for a node's location and utilizes geometric probability to detect replicated nodes. This protocol has modest communication and memory requirements. Furthermore, our solutions apply equally well to any class of network in which the adversary can capture, replicate and insert additional nodes. Examples include wireless ad hoc networks and peer-to-peer networks. We argue that such networks require the resiliency of emergent security techniques to resist an adversary that can subvert an arbitrary number of nodes at unpredictable locations. We expect that distributed algorithms based on emergent properties will provide the best defenses for attacks against these systems.

In the following section, we provide a more detailed description of the node replication attack that we plan to thwart, and we supply a summary of notation used throughout the paper. Then, in Section 3 we summarize some of the earlier proposals and explain why they fail to prevent replication attacks. After discussing some preliminary approaches to distributed detection in Section 4, we present and analyze our two primary protocols, Randomized Multicast and Line-Selected Multicast, in Sections 5 and 6 respectively. We compare and contrast the protocols, discuss synchronization and authentication issues and generalize our

[1]To prevent the adversary from using the location information to find and disable nodes, we could instead broadcast a locator unique to the node's neighborhood that would reveal less information but still be verifiable by the neighbors. For example, the locator could consist of the node's list of neighbors. If the list becomes prohibitively long, each node can broadcast the list to its neighbors but sign a hash of the list. The neighbors verify that they are on the list, check the hash, and then only propagate the hash value, instead of the entire list.

algorithms in Section 8. Finally, we review related research in Section 8.6 and present our future work and conclusions in Sections 9 and 10.

# Chapter 2

# Background

## 2.1 Goals

For a given sensor network, we would like to detect a **node replication attack**, i.e., an attempt by the adversary to add one or more nodes to the network that use the same ID as another node in the network. Ideally, we would like to detect this behavior without centralized monitoring, since centralized solutions suffer from several inherent drawbacks (see Section 3.1). The scheme should also revoke the replicated nodes, so that non-faulty nodes in the network cease to communicate with any nodes injected in this fashion.

We evaluate each protocol's security by examining the probability of detecting an attack given that the adversary inserts $L$ replicas of a subverted node. The protocol must provide robust detection even if the adversary captures additional nodes. We also evaluate the efficiency of each protocol. In a sensor network, communication (both sending and receiving) requires at least an order of magnitude more power than any other operation [14], so our first priority must be minimizing communication, both for the network as a whole and for the individual nodes (since hotspots will quickly exhaust a node's power supply). Moreover, sensor nodes typically have a limited amount of memory, often on the order of a few kilobytes [14]. Thus, any protocol requiring a large amount of memory will be impractical.

## 2.2 Sensor Network Environments

A sensor network typically consists of hundreds, or even thousands, of small, low-cost nodes distributed over a wide area. The nodes are expected to function in an unsupervised fashion even if new nodes are added, or old nodes disappear (e.g., due to power loss or accidental damage). While some networks include a central location for data collection, many operate in an entirely distributed manner, allowing the operators to retrieve aggregated data from any of the nodes in the network. Furthermore, data collection may only occur at irregular intervals. For example, many military applications strive to avoid any centralized and fixed points of failure. Instead, data is collected by mobile units (e.g., unmanned aerial units, foot soldiers, etc.) that access the sensor network at unpredictable locations and utilize the first sensor node they encounter as a conduit for the information accumulated by the network. Since these networks often operate in an unsupervised fashion for long periods of time, we would like to detect a node replication attack soon after it occurs. If we wait until the next data collection cycle, the adversary has time to use its presence in the network to corrupt data, decommission legitimate nodes, or otherwise subvert the network's intended purpose.

We also assume that the adversary cannot readily create new IDs for nodes. Newsome et al. describe several techniques to prevent the adversary from deploying nodes with arbitrary IDs [27]. For example, we can tie each node's ID to the unique knowledge it possesses. If the network uses a key predistribution scheme [6, 13], then a node's ID could correspond to the set of secret keys it shares with its neighbors (e.g., a node's ID is given by the hash of its secret keys). In this system, an adversary gains little advantage by claiming to possess an ID without actually holding the appropriate keys. Assuming the sensor network implements this safeguard, an adversary cannot create a new ID without guessing the appropriate keys (for most systems, this is infeasible), so instead the adversary must capture and clone a legitimate node.

## 2.3 Adversary Model

In examining the security of a sensor network, we take a conservative approach by assuming that the adversary has the ability to surreptitiously capture a limited number of legitimate sensor nodes. We limit the percentage of nodes captured, since an adversary that can capture most or all of the nodes in the network can obviously subvert any protocol running in the network.

Having captured these nodes, the adversary can employ arbitrary attacks on the nodes to extract their private information. For example, the adversary might exploit the unshielded nature of the nodes to read their cryptographic information from memory. The adversary could then clone the node by loading the node's cryptographic information onto multiple generic sensor nodes. Since sensor networks are inherently designed to facilitate ad hoc deployment, these clones can then be easily inserted into arbitrary locations within the network, subject only to the constraint that each inserted node shares at least one key with some of its neighbors. We allow all of the nodes under the adversary's control to communicate and collaborate, but we make the simplifying assumption that any cloned node has at least one legitimate node as a neighbor. In Section 8.4, we show how we can remove this assumption while retaining security. We assume that the adversary operates in a stealthy manner, attempting to avoid detection, since detection could trigger an automated protocol to sweep the network, using a technique such as SWATT [32] to remove compromised nodes, or draw human attention and/or intervention. In the following discussion, we will also assume that nodes under the adversary's control (both the subverted nodes and their clones) continue to follow the protocols described. This allows us to focus on the details of the protocols, but in Section 9, we will suggest methods for relaxing this assumption.

As described above, our adversary model differs from the Dolev-Yao adversary [9] in several respects. Traditionally used to analyze cryptographic protocols, the Dolev-Yao model allows the adversary to read and write messages at any location within the network. However, in our discussion, we restrict the adversary to read and write messages using only the nodes under its control. On the other hand, our model also allows the adversary to subvert and replicate existing nodes in an adaptive manner, capabilities not available to the Dolev-Yao adversary. These capabilities allow the adversary to modify both the network topology and the "trust" topology, since the set of legitimate nodes changes as the adversary subverts nodes and inserts additional replicas.

## 2.4 Notation

For clarity, we list the symbols and notation used throughout the paper below:

| | |
|---|---|
| $n$ | Number of nodes in the network |
| $d$ | Average degree of each node |
| $p$ | Probability a neighbor will replicate location information |
| $g$ | Number of witnesses selected by each neighbor |
| $l_\alpha$ | Location node $\alpha$ claims to occupy |
| $H(M)$ | Hash of M |
| $K_\alpha$ | $\alpha$'s public key |
| $K_\alpha^{-1}$ | $\alpha$'s private key |
| $\{M\}_{K_\alpha^{-1}}$ | $\alpha$'s signature on M |
| $S$ | Set of all possible node IDs |

# Chapter 3

# Previous Protocols

Thus far, protocols for detecting node replication have relied on a trusted base station to provide global detection. For the sake of completeness, we also discuss the use of localized voting mechanisms. We consider these protocols in the abstract; for specific examples of previous protocols, see Section 8.6. Until now, it was generally believed that these two alternatives exhausted the space of possibilities. This paper expands the design space to offer new alternatives with strong security and efficiency characteristics.

## 3.1 Centralized Detection

The most straightforward detection scheme requires each node to send a list of its neighbors and their claimed locations to the base station. The base station can then examine every neighbor list to look for replicated nodes. If it discovers one or more replicas, it can revoke the replicated nodes by flooding the network with an authenticated revocation message.

While conceptually simple, this approach suffers from several drawbacks inherent in a centralized system. First, the base station becomes a single point of failure. Any compromise of the base station or the communication channel around the base station will render this protocol useless. Furthermore, the nodes closest to the base station will receive the brunt of the routing load and will become attractive targets for the adversary. The protocol also delays revocation, since the base station must wait for all of the reports to come in, analyze them for conflicts and then flood revocations throughout the network. A distributed or local protocol could potentially revoke replicated nodes in a more timely fashion. Finally, many networks do not have the luxury of a powerful base station, making a distributed solution a necessity.

In terms of security, this protocol achieves 100% detection of all replicated nodes, assuming all messages successfully reach the base station. As far as efficiency, if we assume that the average path length[1] to the base station is $O(\sqrt{n})$ and each node has an average degree $d$ (for $d \ll n$), then this protocol requires $O(n\sqrt{n})$ communication for all of the reports from the nodes to reach the base station. The storage required at each node is $O(d)$. At the base station, the protocol requires $O(n \cdot d)$, though storage is presumably less of a concern for the base station.

## 3.2 Local Detection

To avoid relying on a central base station, we could instead rely on a node's neighbors to perform replication detection. Using a voting mechanism, the neighbors can reach a consensus on the legitimacy of a given node. Unfortunately, while achieving detection in a distributed fashion, this method fails to detect distributed node replication in disjoint neighborhoods within the network. As long as the replicated nodes are at least two hops away from each other, a purely local approach cannot succeed.

[1]This will hold true if the sensor network deployment approximates any regular polygon.

# Chapter 4

# Preliminary Approaches

One might imagine addressing the shortcomings of previously proposed protocols by implementing distributed detection using a simple broadcast scheme, or by using deterministic replication of location claims. To the best of our knowledge, neither of these protocols have been discussed in the literature. Despite their drawbacks, we discuss them to provide background and intuition for our two primary protocols, Randomized Multicast and Line-Selected Multicast, presented in Sections 5 and 6 respectively. In all four protocols, we assume that nodes know their own geographic positions. Numerous researchers have proposed schemes for determining node location, using everything from highly abstract graph embeddings [28], to connectivity information [8], to powerful beacon nodes placed on the perimeter of the network [5]. Some of these proposals require that some or all of the nodes have GPS receivers, but many do not. For our purposes, any of these protocols will suffice. We also assume that the nodes in the network remain relatively stationary, at least for the time it takes to perform one round of replication detection. If the network designers anticipate occasional mobility, they can schedule regular detection rounds. As long as a node successfully participates in a round, it can continue to communicate until the next round, even if its position changes in the interim. We discuss additional timing details in Section 8.2.

## 4.1 Node-To-Network Broadcasting

One approach to distributed detection utilizes a simple broadcast protocol. Essentially, each node in the network uses an authenticated broadcast message to flood the network with its location information. Each node stores the location information for its neighbors and if it receives a conflicting claim, revokes the offending node.

This protocol achieves 100% detection of all duplicate location claims under the assumption that the broadcasts reach every node. This assumption may not hold if the adversary can jam key areas or otherwise interfere with communication paths through the network. Nodes could employ redundant messages or authenticated acknowledgment techniques to try to thwart such an attack. In terms of efficiency, this protocol requires each node to store location information about its $d$ neighbors. One node's location broadcast requires $O(n)$ messages, assuming the nodes employ a duplicate suppression algorithm in which each node only broadcasts a given message once. Thus, the total communication cost for the protocol is $O(n^2)$. Given the simplicity of the scheme and the level of security achieved, this cost may be justifiable for small networks. However, for large networks, the $n^2$ factor is too costly, so we investigate schemes with a lower cost.

## 4.2 Deterministic Multicast

To improve on the communication cost of the previous protocol, we describe a detection protocol that only shares a node's location claim with a limited subset of deterministically chosen "witness" nodes. When a node broadcasts its location claim, its neighbors forward that claim to a subset of the nodes called witnesses. The witnesses are chosen as a function of the node's ID. If the adversary replicates a node, the witnesses will receive two different location claims for the same node ID. The conflicting location claims become evidence to trigger the revocation of the replicated node.

More formally, in this protocol, whenever node $\gamma$ hears a location claim $l_\alpha$ from node $\alpha$, it computes $F(\alpha) = \{\omega_1, \omega_2, \ldots, \omega_g\}$, where $F$ maps each node ID in the set of possible node IDs, $S$, to a set of $g$ node IDs:

$$F : S \to \{\sigma : \sigma \in 2^S, |\sigma| = g\} \tag{4.1}$$

The nodes with IDs in the set $\{\omega_1, \omega_2, \ldots, \omega_g\}$ constitute the witnesses for node $\alpha$. Node $\gamma$ forwards $l_\alpha$ to each of these witnesses. If $\alpha$ claims to be at more than one location, the witnesses will receive conflicting location claims, which they can flood through the network, discrediting $\alpha$.

In this protocol, each node in the network stores $g$ location claims on average. For communication, assuming $\alpha$'s neighbors do not collaborate, we will need each of $\alpha$'s neighbors to probabilistically decide which of the $\omega_i$ to inform. If each node selects $\frac{g \ln g}{d}$ random destinations from the set of possible $\omega_i$, then the coupon collector's problem [7] assures us that each of the $\omega_i$'s will receive at least one of the location claims. Assuming an average network path length of $O(\sqrt{n})$ nodes, this results in $O(\frac{g \ln g \sqrt{n}}{d})$ messages. Unfortunately, this cost does not provide much security. Since $F$ is a deterministic function, an adversary can also determine the $\omega_i$s. Thus, they become targets for subversion. If the adversary can capture or jam all $g$ of the messages destined to the $\omega_i$s, then she can create as many replicas of $\alpha$ as she desires (limited only by the requirement that no two replicas share a neighbor). Since the communication costs of this protocol grow as $O(g \ln g)$, we cannot afford a large value for $g$, and yet a small value for $g$ allows the adversary almost unlimited replication abilities after compromising a fixed number of nodes; in other words, if the adversary controls the $g$ witnesses for $\alpha$, she can create unlimited replicas of $\alpha$ and suppress the conflicting reports arriving at the witness nodes. These disadvantages make this protocol unappealing.

# Chapter 5

# Randomized Multicast

To improve the resiliency of the deterministic multicast protocol discussed in Section 4.2, we propose a new protocol that randomizes the witnesses for a given node's location claim, so that the adversary cannot anticipate their identities. When a node announces its location, each of its neighbors sends a copy of the location claim to a set of randomly selected witness nodes. If the adversary replicates a node, then two sets of witnesses will be selected. In a network of $n$ nodes, if each location produces $\sqrt{n}$ witnesses, then the birthday paradox predicts at least one collision with high probability, i.e., at least one witness will receive a pair of conflicting location claims. The two conflicting locations claims form sufficient evidence to revoke the node, so the witness can flood the pair of locations claims through the network, and each node can independently confirm the revocation decision.

## 5.1 Assumptions

As discussed in Section 4, our protocols assume that each node knows its own location. We also assume that the network utilizes an identity-based public key system such that each node $\alpha$ is deployed with a private key, $K_\alpha^{-1}$, and any other node can calculate $\alpha$'s public key using $\alpha$'s ID, i.e., $K_\alpha = f(\alpha)$. If necessary, we could replace this system with a more traditional PKI in which we assume the network authorities use a master public/private-key pair ($K_M$, $K_M^{-1}$) to sign $\alpha$'s public key; however, transmitting this public-key certificate will have a substantial communication overhead.

Traditionally, researchers have assumed that public key systems exceed the memory and computational capacity of sensor nodes. However, public key cryptography on new sensor hardware may not be as prohibitive as traditionally assumed. In recent work, Malan et al. demonstrate that they can successfully generate 163 bit ECC keys on the MICA 2 in under 34 seconds [22]. Furthermore, the latest generation of Telos sensors come with 10KB of RAM and can achieve 5x the data rate of the MICA 2, making public-key algorithms more practical. In Section 8.3.2, we discuss how we could instead use symmetric-key cryptography to lower the computational overhead, at the expense of additional communication.

## 5.2 Protocol Description

At a high level, the protocol has each node broadcast its location claim, along with a signature authenticating the claim. Each of the node's neighbors probabilistically forwards the claim to a randomly selected set of witness nodes. If any witness receives two different location claims for the same node ID, it can revoke the replicated node. The birthday paradox ensures that we detect replication with high probability using a relatively limited number of witnesses.

More formally, each node $\alpha$ broadcasts a location claim to its neighbors, $\beta_1$, $\beta_2$, ..., $\beta_d$. The location claim has the format $\langle ID_\alpha, l_\alpha, \{H(ID_\alpha, l_\alpha)\}_{K_\alpha^{-1}}\rangle$, where $l_\alpha$ represents $\alpha$'s location (e.g., geographic coordinates $(x, y)$). Upon hearing this announcement, each neighbor, $\beta_i$, verifies $\alpha$'s signature and the plausibility of $l_\alpha$ (for example, if each node knows its own position and has some knowledge of the maximum propagation radius of the communication layer, then it can loosely bound $\alpha$'s set of potential locations). Then, with probability $p$, each neighbor selects $g$ random locations within the network and uses geographic routing (e.g., GPSR [19]) to forward $\alpha$'s claim to the nodes closest to the chosen locations (as in GHT [30]). Since we have assumed the nodes are distributed randomly, this should produce a random selection from the nodes in the network. In Section 5.3, we show that the probability of selecting the same node more than once is generally negligible. Collectively, the nodes chosen by the neighbors constitute the witnesses for $\alpha$.

Each witness that receives a location claim, first verifies the signature. Then, it checks the ID against all of the location claims it has received thus far. If it ever receives two different locations claims for the same node ID, then it has detected a node replication attack, and these two location claims serve as evidence to revoke the node. It blacklists $\alpha$ from further communication by immediately flooding the network with the pair of conflicting location claims, $l_\alpha$ and $l'_\alpha$. Each node receiving this pair can independently verify the signatures and agree with the revocation decision. Thus, the sensor network both detects and defeats the node replication attack in a fully distributed manner. Furthermore, the randomization prevents the adversary from predicting which node will detect the replication.

## 5.3 Security Analysis

Let malicious node $\alpha$ claim to be at $L$ locations, $l_1, l_2, \ldots, l_L$. We would like to determine the probability of a collision using the randomized multicast protocol outlined above, since a collision at a witness corresponds to detection of $\alpha$'s replication. At each location $l_i$, $p \cdot d$ nodes randomly select $g$ witnesses. If the neighbors coordinated perfectly, this would store $\alpha$'s location claim at exactly $p \cdot d \cdot g$ locations. However, since we prefer to have each neighbor act independently, there may be some amount of overlap between the witnesses each neighbor selects. To determine the impact of this overlap, we would like to determine the number of nodes, $N_{receive}$, that will receive the location claim assuming the neighbors choose witnesses independently. If $P_{claim}$ is the probability that a node hears at least one claim and $P_{none}$ is the probability that a node hears no location claims, then we have:

$$E[N_{receive}] = n \cdot P_{claim} \tag{5.1}$$

$$P_{claim} = 1 - P_{none} \tag{5.2}$$

Since each neighbor is assumed to select $g$ random, unique witness locations, the probability $(P_f)$ that a node fails to hear any of the $g$ announcements from one neighbor is:

$$P_f = 1 - \frac{g}{n} \tag{5.3}$$

Since each neighbor decides independently whether to send out location claims, the number of nodes that actually send out location claims is distributed binomially, with mean $p \cdot d$ and variance $d \cdot p(1-p)$. For a network with $d = 20$ and $p = \frac{1}{10}$, the variance will be less than 0.005, so we will approximate the number of neighbors that send out locations claims as $p \cdot d$. Since the neighbors choose their destinations independently, we have:

$$P_{none} = \left(1 - \frac{g}{n}\right)^{p \cdot d} \tag{5.4}$$

Combining equations 5.1, 5.2 and 5.4, the number of witness nodes that receive at least one location claim is:

$$E[N_{receive}] = n \cdot \left(1 - \left(1 - \frac{g}{n}\right)^{p \cdot d}\right) \tag{5.5}$$

The Binomial Theorem allows us to approximate $(1 - x)^y$ as $(1 - xy)$ for small $x$, so as long as $g \ll n$, we have $N_{receive} \approx p \cdot d \cdot g$, so overlapping witness locations should not impact the security of the protocol. As an example, in a network with $n = 10,000$, $g = 100$, $d = 20$, and $p = 0.1$, perfect coordination would tell 200 nodes, while independent selection would tell 199. Thus, for the remainder of the analysis, we will assume that $p \cdot d \cdot g$ nodes receive each location claim.

If the adversary inserts $L$ replicas of $\alpha$, we would like to determine the probability that two conflicting location reports collide at some witness node, since this corresponds to the probability that a witness detects the node replication. Note that even if there are more than two replicas of $\alpha$, we still only need two location claims to collide in order to completely revoke all $L$ of the replicas, since one collision will prompt a network-wide flood of the duplicate claims ($l_i$ and $l_j$) and any other node that has heard location claim $l_k$ for $k \neq i, j$ will also revoke $\alpha$.

Following the standard derivation of the birthday paradox [7], the probability $P_{nc_1}$ that the $p \cdot d \cdot g$ recipients of claim $l_1$ do not receive any of the $p \cdot d \cdot g$ copies of claim $l_2$ is given by:

$$P_{nc_1} = \left(1 - \frac{p \cdot d \cdot g}{n}\right)^{p \cdot d \cdot g} \tag{5.6}$$

Similarly, the probability $P_{nc_2}$ that the $2 \cdot p \cdot d \cdot g$ recipients of claims $l_1$ and $l_2$ do not receive any of the $p \cdot d \cdot g$ copies of claim $l_3$ is given by:

$$P_{nc_2} = \left(1 - \frac{2 \cdot p \cdot d \cdot g}{n}\right)^{p \cdot d \cdot g} \tag{5.7}$$

Thus, the probability $P_{nc}$ of no collisions at all is given by:

$$P_{nc} = \prod_{i=1}^{L-1} \left(1 - \frac{i \cdot p \cdot d \cdot g}{n}\right)^{p \cdot d \cdot g} \tag{5.8}$$

Using the standard approximation that $(1+x) \leq e^{x}$, we have:

$$P_{nc} \leq \prod_{i=1}^{L-1} e^{\frac{-i \cdot p^2 \cdot d^2 \cdot g^2}{n}} \tag{5.9}$$

$$\leq e^{\frac{-p^2 \cdot d^2 \cdot g^2}{n} \sum_{i=1}^{L-1} i} \tag{5.10}$$

$$\leq e^{\frac{-p^2 \cdot d^2 \cdot g^2}{n} \frac{L(L-1)}{2}} \tag{5.11}$$

Since the probability of a collision, $P_c$, is simply $1 - P_{nc}$, the probability of detecting $L$ replicas is:

$$P_c \geq 1 - e^{\frac{-p^2 \cdot d^2 \cdot g^2}{n} \frac{L(L-1)}{2}} \tag{5.12}$$

Thus, if $n = 10,000$, $g = 100$, $d = 20$, and $p = 0.05$, we will detect a single replication of $\alpha$ with probability greater than 63%, and if $\alpha$ is replicated twice, we will detect it with probability greater than 95%.

Unlike the deterministic proposal (Section 4.2), we no longer need to worry about the adversary using a limited number of captured nodes to enable an unlimited amount of replication. If the adversary captures neighboring nodes $\alpha$ and $\beta$, then the total number of claims about either node is reduced by $\frac{1}{d}$, essentially reducing the neighbor count of each node by one. Since all of the protocol decisions are made locally by individual nodes, the adversary has only two options remaining: it can disrupt the routing of packets from the remaining legitimate neighbors or it can capture all of the legitimate neighbors. Routing disruptions create tell-tale signs of the adversary's presence in the network and will be avoided by a prudent adversary. Capturing all of the neighbors of a node targeted for replication leads to a practical attack which we address in Section 8.4.

## 5.4 Efficiency Analysis

This scheme still poses a relatively high storage cost. On average, each node will have to store $p \cdot d \cdot g$ location claims. To ensure a collision with greater than 50% probability, $p \cdot d \cdot g$ will have to be on the order of $O(\sqrt{n})$. Even if we can reduce the size of each claim to the payload of a packet, 36B, our hypothetical network with $n = 10,000$, $g = 100$, $d = 20$, and $p = 0.05$ will require, on average, that each node store 3,700 B, which is just under 91% of the Mica 2's total RAM. Similarly, the communication requirements of the protocol are non-trivial. For each node, we generate $p \cdot d \cdot g$ messages that must be evenly spread throughout the network. In a network randomly deployed on the unit square, the average distance between any two randomly chosen nodes is approximately $0.521\sqrt{n} \approx \frac{\sqrt{n}}{2}$, so the communication costs for the network are on the order of $O(n\sqrt{n} \cdot p \cdot d \cdot g)$. As mentioned before, $p \cdot d \cdot g \approx O(\sqrt{n})$, so our communication costs are $O(n^2)$, equivalent to those of the naive broadcast scheme outlined in Section 4.1.

We can employ a number of enhancements to improve both the communication and space requirements of our protocol. First, we can trade resiliency for memory and communication savings. For example, in our hypothetical network[1], if we are willing to allow the adversary to create up to four replicas of $\alpha$, then we can reduce the number of messages sent out, $g$, by $75\%$. Since communication costs are $O(g^2)$ we save on communication and require less than 1KB of space, but we would still detect the adversary's presence with probability greater than $50\%$. We can also save both communication and space by reducing the number of location claims sent out by $\frac{1}{d}$, i.e. $g' = \frac{g}{d}$. Each recipient of one of these claims uses a broadcast message to query her neighbors as to whether they have a conflicting value. Even with these additional queries, the communication costs are now $O(\sqrt{n} \cdot p \cdot g)$ messages per node. Finally, we can reduce the storage burden by introducing a loose notion of synchronization into the process – see Section 8.2.

[1]The network in which $n = 10,000$, $g = 100$, $d = 20$, and $p = 0.05$.

# Chapter 6

# Line-Selected Multicast

## 6.1 Overview

To reduce the communication costs of our randomized multicast protocol, we investigate a different scheme to detect conflicting location claims. Inspired by Braginsky and Estrin's work on Rumor Routing [4], we note that nodes in a sensor network function both as sensing units and as routers. For a location claim to travel from node $\alpha$ to node $\gamma$, it must pass through several intermediate nodes as well. If these intermediate nodes also store the location claim, then we have effectively drawn a line across the network. If a conflicting location claim ever crosses the line, then the node at the intersection will detect the conflict and initiate a revocation broadcast. Since the expected number of intersections, $c$, of $x$ randomly drawn lines[1] intersecting within the bounds of the unit circle is given by:

$$E(c) = x(x-1)\left(\frac{1}{6} + \frac{245}{144\pi^2}\right) \tag{6.1}$$

we only need a few such lines to insure an intersection. For example, with only three lines, we expect two collisions (see Solomon's lecture notes [34] for details of the derivation). With this insight in mind, we can craft an alternate detection protocol with improved performance.

[1]Lines drawn by randomly selecting two points, $p_1$ and $p_2$, from within the unit circle, and drawing the line extending through them.

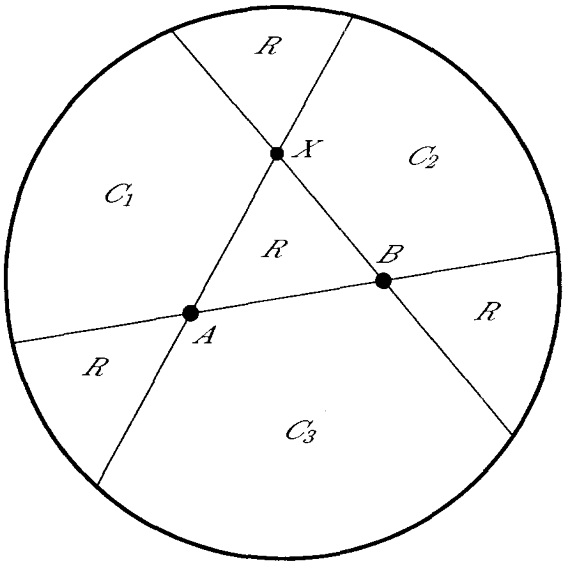


Figure 6.1: **Line-Segment Intersection** *Given three randomly selected points, $A, B$ and $X$, then if the fourth randomly-selected point $Y$ falls in any of the $C_i$ regions, the resulting quadrilateral will be convex. If $Y$ falls in any of the $R$ regions, the figure will be re-entrant. For $\overline{XY}$ to intersect $\overline{AB}$, $Y$ must fall in region $C_3$.*

## 6.2 Protocol Outline

Our new protocol modifies the Randomized Multicast protocol, so that we fix $p \cdot d \cdot g$ as a small constant $r$. When $\alpha$'s neighbors send out the evidence of $\alpha$'s location claim to the $r$ witnesses, each of the nodes along the route stores a copy of the location claim as well. For example, let $\beta_i$ send a copy of $\alpha$'s location claim $l_\alpha$ to $\gamma_j$ via $\sigma_1, \sigma_2 \ldots \sigma_m$. Upon receiving $l_\alpha$, $\sigma_k$ verifies the signature on the claim, checks for a conflict with the claims already in its buffer, stores a copy of $l_\alpha$ in its buffer, and then forwards $l_\alpha$ to $\sigma_{k+1}$. If any of the nodes discovers a conflict, i.e., finds another location claim $l'$ for $\alpha$ such that $l_\alpha \neq l'_\alpha$, then it floods the network with the unforgeable evidence (the conflicting set of signed location claims) of $\alpha$'s attempted replication, resulting in a distributed revocation of $\alpha$. If the collision happens to occur at a replica, it still does not preclude another collision from occurring elsewhere in the network. Also, since all protocol decisions are made locally and probabilistically, the adversary cannot predict the location of the collision, so the probability of a collision occurring at a node under the adversary's control will be negligible, unless the adversary already has an overwhelming presence within the network.

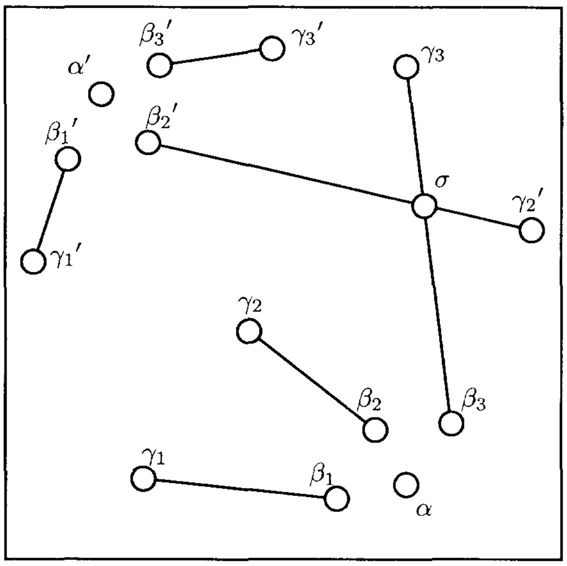


Figure 6.2: **Line-Selected Multicast** *In this figure, the adversary has created a replica of $\alpha$, $\alpha'$. Neighbors ($\beta_i$ and $\beta_i'$) at these locations all report these claims to randomly selected witnesses $\gamma_i$ and $\gamma_i'$, which results in an intersection at $\sigma$.*

## 6.3 Analysis

As described above, our protocol actually "draws" line-segments, not lines, through the network. Unfortunately, the probability of two segments intersecting is considerably less than that of two lines intersecting (given above by Equation 6.1). To find the probability that two line-segments intersect[2], we can use the solution to Sylvester's Four-Point Problem. The Four-Point Problem asks for the probability that four randomly selected points in a convex domain will form a re-entrant quadrilateral, i.e., one in which one point falls within the triangle formed by the other three points. Solomon shows that if the points are selected from a circular domain, then the probability that the points form a re-entrant quadrilateral is $\frac{35}{12\pi^2}$ [34]. If we select our first three points at random, then we can divide the region into seven sections: four re-entrant and three convex (see Figure 6.1). The two line-segments will only intersect if the fourth point falls in the convex region $C_3$. Thus, the probability of intersection is given by:

$$P_{intersect} = \frac{1}{3}\left(1 - \frac{35}{12\pi^2}\right) \approx 0.235 \tag{6.2}$$

[2]Segments drawn by randomly selecting two points, $p_1$ and $p_2$, from within the unit circle, and drawing the line segment connecting them.

To further complicate the analysis, our line segments are not drawn independently at random, but originate from a central point and radiate out in random directions (see Figure 6.2). However, Monte-Carlo simulations indicate that even if we only draw two random segments originating from $\alpha$ and two from $\alpha'$, the probability of at least one intersection is greater than 56%, and with five line segments per point, we have a 95% probability of intersection. Since an intersection corresponds to detecting a node replication attack, we can detect an attack with high probability, using only a constant number of line-segments. While this approach depends on the routing topology of the network, our simulations indicate that it reliably detects node replication in realistic scenarios (see Section 7).

Since we only use a constant number of line-segments per node, the Line-Selected Multicast protocol has very reasonable performance characteristics. Assuming each line-segment is of length $O(\sqrt{n})$, then our protocol only requires $O(n\sqrt{n})$ communication for the entire network and each node stores $O(\sqrt{n})$ location claims. We can also reduce the storage requirement by using the time synchronization enhancements described in Section 8.2.

# Chapter 7

# Simulations

To verify the accuracy of our theoretical predictions, we ran simulations to measure the communication requirements of our two primary protocols, Randomized Multicast and Line-Selected Multicast. Since Line-Selected Multicast relies on the topology of the network to detect node replications, we also evaluated its detection rate in a variety of network configurations. Examples of the network topologies tested appear in Appendix A.

In our simulations, we deploy $n$ nodes uniformly at random within a $500 \times 500$ square, with $n$ varying between 1,000 and 10,000. We assume the standard unit-disc bidirectional communication model and we adjust the communication range, so that each node will have approximately 40 neighbors on average[1]. We use an average of the total number of messages sent or received per node as a measure of the communication requirements, and we measure resiliency by counting the number of times we must run the protocol in order to detect a single node replication (i.e., we select a random node and insert one replica into the network). Thus, we calculate the probability of detection, $P_d$ as:

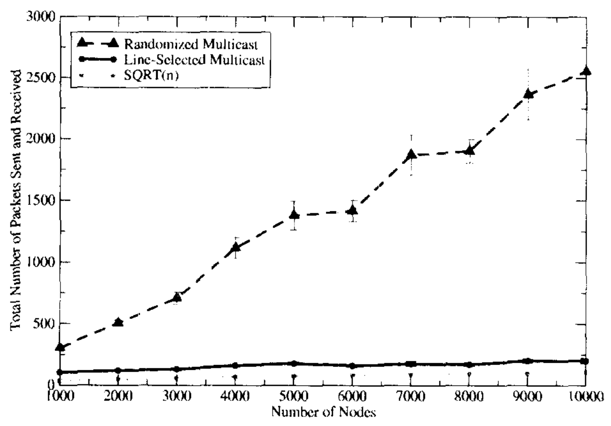
$$P_d = \frac{1}{\textit{\# repetitions}} \tag{7.1}$$

For the Randomized Multicast protocol, we used

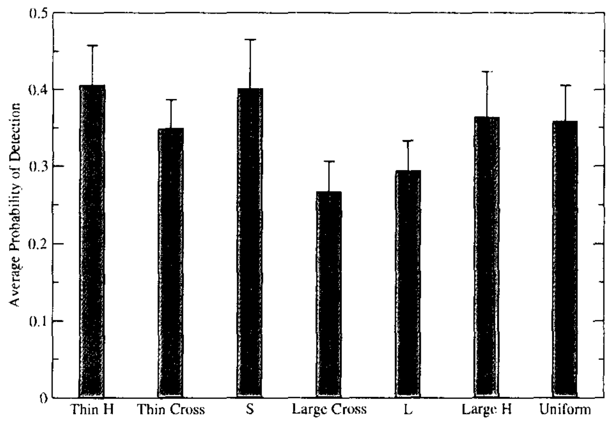
$$p \cdot d \cdot g = \sqrt{n}$$

[1]Our simulations show little variance with values of $d$ ranging from 10 to 50, though the communication required by the Randomized Multicast drops slightly for larger values of $d$.

which theoretically gives us a 63% probability to detect replication, and for Line-Selected Multicast we used $r = 6$ (i.e., each location claim creates six line segments). For each network configuration, we generated twenty random graphs and averaged the results of ten trials on each graph.



(a) **Communication Overhead** *This figure indicates the average amount of communication per node. Note that Randomized Multicast scales linearly with the number of nodes, while Line-Selected Multicast scales as $\sqrt{n}$. The error bars on the data from the two Multicast schemes represent the standard error.*



(b) **Probability of Detection** *This graph illustrates the average probability of detecting a single node replication using Line-Selected Multicast in a variety of topologies. See Appendix A for examples of the graphs tested.*

Figure 7.1: **Simulation Results**

As shown in Figure 7.1(a), the simulations closely match our theoretical predictions. The communication for Randomized Multicast scales linearly with the number of nodes, while the Line-Selected Multicast only grows at $O(\sqrt{n})$. Our simulations also indicate that the maximum amount of communication required of any one node as compared to the average case scales logarithmically with the number of nodes in the graph. Using Randomized Multicast in a network with 1,000 nodes requires a maximum of four times as much communication as the average case, and with 10,000 nodes, it requires seven times as much. With Line-Selected Multicast, the maximum amount of communication in a network with 1,000 nodes is twice as much as the average case and four times as much for 10,000 nodes.

As Figure 7.1(b) illustrates, Line-Selected Multicast reliably detects node replications in a variety of irregular network configurations. To improve the probability of detection, we can always repeat the protocol or add a few additional line segments per node.

# Chapter 8

# Discussion

In this section, we compare the performance of the various protocols we have discussed, and we present several techniques based on loose-time synchronization that reduce the storage requirements of the protocols. We also discuss potential issues that may arise from our use of public-key cryptography, as well as symmetric alternatives that would require less computational overhead at the price of additional communication. In Section 8.4, we describe a sophisticated attack that applies to all of the protocols we have discussed, and we present a defense against it. Finally, based on the success of our Randomized Multicast and Line-Selected Multicast protocols, we argue that algorithms based on emergent properties offer the most promising techniques for providing security in sensor networks.

## 8.1 Protocol Comparison

Table 8.1 summarizes the costs for each of the distributed protocols previously discussed. The Broadcast protocol offers the simplest solution, but the communication overhead will only be tolerable for small net-

| | *Communication* | *Memory* |
|---|---|---|
| Broadcast | $O(n^2)$ | $O(d)$ |
| Deterministic Multicast | $O(\frac{g \ln g \sqrt{n}}{d})$ | $O(g)$ |
| Randomized Multicast | $O(n^2)$ | $O(\sqrt{n})$ |
| Line-Selected Multicast | $O(n\sqrt{n})$ | $O(\sqrt{n})$ |

Table 8.1: **Summary of Protocol Costs** *This tables illustrates the memory and communication costs for each protocol. The communication costs are for the entire network and the memory costs are per node. The Line-Selected Multicast protocol offers the most efficient solution in terms of resiliency versus cost.*

works. Deterministic Multicast improves on the communication requirements, but by selecting a fixed set of witnesses, it loses resiliency. An attacker can perform unlimited replications after only compromising a fixed number of nodes. Randomized Multicast provides excellent resiliency, since it prevents the adversary from anticipating the identity of the witnesses. Unfortunately, it imposes communication overhead equal to that of the Broadcast scheme. However, for networks in which the number of nodes is less than the square of the average degree, Randomized-Multicast will tend to be more space efficient. Finally, Line-Selected Multicast uses less communication than Broadcast or Randomized Multicast, but provides comparable or greater resiliency, making it a particularly attractive choice. Our simulations confirm that this resiliency remains in a variety of network configurations (see Section 7). We can also reduce the storage requirements for these protocols by using the time synchronization enhancements described in Section 8.2.

## 8.2 Synchronized Detection

We now consider the synchronization issues involved in detecting node replication. All of the protocols described above require a loose notion of synchronization to insure timely detection. Various protocols exist that can offer the coarse-grained level of synchronization that we require. For example, Reference-Broadcast Synchronization (RBS) enables pairwise synchronization with low overhead and high precision [12]. Hu and Servetto describe a protocol that provides asymptotically optimal time synchronization in dense networks [18]. Any of these protocols will suffice for our purposes.

Deciding how often to perform the detection protocol trades efficiency of detection against the communication and storage costs required by each iteration. However, as we describe below, we can leverage our assumption of loose synchronization to mitigate the cost of running the protocols.

### 8.2.1 High Noon

In one modification, detection happens during a fixed window of time (of length $t$) that occurs every $T$ units of time (for $T \gg t$). At the beginning of each time window, each node broadcasts its location claim to its neighbors, who then resend it to random locations in the network. Each node looks for conflicts in location claims arriving during the time window and revokes conflicting nodes. After time $t$ has elapsed, the nodes forget all of the location claims (but continue to remember the list of revoked nodes). Using this modification, the nodes only devote significant memory resources to detection during the brief time window of length $t$. The rest of the time $(T - t)$, they can utilize their entire memory for non-detection purposes.

### 8.2.2 Time Slots

As an alternate approach, we assume the node IDs are randomly distributed on some fixed interval $[0..N]$. For a given protocol parameter $k$, we can divide time into epochs of length $T$, with each epoch consisting of $k$ time slots. During each epoch, in time slot $s$, all of the nodes with IDs located in the interval $[s\cdot\frac{N}{k}..(s+1)\cdot\frac{N}{k}]$ broadcast their location claims to their neighbors, who then follow the standard protocol. Nodes receiving a location claim from a node with an ID in that interval store the claim for the duration of the time slot and check for conflicts. At the end of the time slot, they can forget all of the claims they have heard. Using this method reduces the storage requirements at each node to $O(\frac{p\cdot d\cdot g}{k})$.

### 8.2.3 Security Requirements

Since both of these modifications operate deterministically, the adversary might attempt to launch her replication attack between time slots or refuse to follow the protocol (i.e., by not broadcasting her location at the specified time). However, we can thwart this behavior by having each node remember which neighbors it heard from during the previous epoch. Later, if a node hears from a neighbor that did not participate in the previous epoch, it will refuse to communicate with that node until the node successfully participates in a detection epoch. This effectively precludes the adversary from avoiding randomized detection.[1] Since the nodes in most sensor networks must already remember a list of their neighbors, this would only require an additional $d$ bits. Of greater concern is the fact that this modification limits when new nodes can join the network. This can be mitigated by appropriate choices for $t$, $T$, and $k$, as well as decisions regarding deployment timing.

## 8.3 Authentication

The use of public-key signatures requires several extensions to our protocols to prevent the signatures themselves from becoming a security threat. Also, while we argued in Section 5.1 that algorithmic and hardware improvements are beginning to make public-key cryptography practical for sensor networks, we also present

[1]Technically, both of these modifications would require some additional configuration to account for propagation delays and uncertainty as to the size and width of the network, but the essential idea remains sound.

a mechanism that utilizes symmetric one-time signatures as an alternative to the public-key authentication previously assumed. While the symmetric signatures reduce the computational overhead of generating and verifying signatures, they require additional communication, making them less appealing than the public-key approach.

### 8.3.1 Public Key Security Adjustments

To prevent the use of signatures from becoming a security liability, we need to ensure that the adversary cannot perform a Denial-of-Service attack by making nodes verify bogus signatures or by reporting its neighbors' position claims to every node in the network, rather than probabilistically reporting them to $g$ nodes. In practice, these two attacks are unlikely to be a concern. Since the basic protocol currently requires each node that receives a location claim to verify the signatures contained within the claim, legitimate nodes will immediately detect any attempt to inject faulty signatures. If we add some form of neighbor-to-neighbor authentication, then a node identifying a faulty signature can also assign appropriate blame for the fault. It may then choose to blacklist or otherwise revoke the guilty party. If the adversary can flood a node with faulty signatures not originating from a node (via some other broadcast source), then the adversary could just as easily perform a jamming attack or otherwise interfere with legitimate communication. Neighbor-to-neighbor authentication also prevents the adversary from framing an innocent third party.

To address the second concern, we note that when a legitimate node, $\gamma$, forwards a location claim to its $g$ randomly chosen locations, these location claims must be routed through its neighbors. If the neighbors listen promiscuously, they can all detect $\gamma$'s attempts to forward the same location claim more than $g$ times and refuse to forward additional claims. Unfortunately, this does require each node to keep a count associated with each neighbor, but it may be necessary to prevent the adversary from wasting the network's collective memory.

### 8.3.2 Symmetric Alternatives

As an alternative to computation-intensive public-key algorithms, researchers have proposed using more efficient symmetric cryptographic mechanisms for sensor networks – for example, broadcast authentication based on one-way chains and time synchronization [20, 29], or one-time signatures based on one-way functions [25].

For the purposes of this paper, if sensor nodes only need to sign a single location statement, a one-time signature will suffice. For example, we could use the Merkle-Winternitz signature [25], which has already been successfully used for stream signatures [31]. To verify a Merkle-Winternitz signature, a verifier only needs to possess an authentic verification value, i.e., the public key, which in this case is a hash value over several one-way chain values. However, since storing all of the public values of all the nodes would have a high overhead, we could instead construct a Merkle hash tree [24] over all public values, and the resulting root node could be used to authenticate any one-time signature in that tree. For signature verification, a node would include all the values in the hash tree that allow a verifier to recompute the root node of the hash tree; for $n$ nodes, this approach would require $\ln n$ extra values. Using the parameters suggested by Rohatgi [31] for the Merkle-Winternitz signature, a signature is 230 bytes large, and the additional verification values would require an additional 100 bytes. This is quite large for a sensor network, and thus we suggest using asymmetric cryptography to achieve smaller messages, despite the higher verification cost. Since message transmission accounts for the majority of energy consumption, we may conserve energy by sending smaller messages but requiring a higher computation overhead.

## 8.4 Masked-Replication Attacks

In our discussion of the resiliency of the various protocols, we have assumed that each node has at least one legitimate neighbor. Without this assumptions, all of the protocols discussed thus far may fail to detect node replication. If the adversary compromises all of $\alpha$'s neighbors, then she can create one replica of $\alpha$ without fear of detection, since the compromised nodes will not send out location claims for the original $\alpha$. To create a second replica of $\alpha$, the adversary must also compromise the nodes surrounding the first replica. Thus, the adversary must compromise an additional $d$ nodes for each replica of $\alpha$ she wishes to create. However, if the compromised nodes mask the replicated nodes, the adversary can improve her foothold in the network. For example, if the adversary compromises nodes $\mu_1, \mu_2 \ldots \mu_k$, then she can assign replicas of $\{\mu_1, \mu_2 \ldots \mu_{i-1}, \mu_{i+1} \ldots \mu_k\}$ as neighbors of $\mu_i$. This gives the adversary the influence of $k^2$ nodes after she compromises $k$ nodes. As a concrete example, suppose the adversary compromises nodes $\mu_1$ and $\mu_2$. Now, she can create replicas $\mu'_1$ and $\mu'_2$. By assigning $\mu_1$ as $\mu'_2$'s only neighbor, and $\mu_2$ as $\mu'_1$'s only neighbor, the compromised nodes can mask the replicas. None of the protocols discussed will detect these replicas, since they rely on a replicated node's neighbors to propagate the replica's location claims.

Fortunately, we can thwart this attack with a straightforward defense. Each node $\beta$ maintains a list of the $m$ nodes from which it has seen the most traffic. For each node $\alpha$ on the list, $\beta$ appoints itself a pseudo-neighbor of $\alpha$ with probability

$$P_{pseudo} \propto \frac{\mathit{traffic}_\alpha}{\mathit{claims}_\alpha} \tag{8.1}$$

where $\mathit{traffic}_\alpha$ is the amount of traffic that $\beta$ has seen from $\alpha$, and $\mathit{claims}_\alpha$ is the number of location claims $\beta$ has seen concerning $\alpha$. As a pseudo-neighbor, $\beta$ requests a location claim directly from $\alpha$. If $\alpha$ fails to respond, $\beta$ ceases to forward traffic from $\alpha$. If $\alpha$ does respond, $\beta$ follows the same protocol-dependent behavior as the real neighbors of $\alpha$ (e.g., in the Randomized Multicast, $\beta$ will forward the location claim to a randomly selected set of witness nodes).

The use of pseudo-neighbors successfully defeats masked-replication attacks. If the masked nodes fail to send out location claims, then the closest legitimate nodes will have a higher probability of appointing themselves pseudo-neighbors. If the replicated nodes fail to respond to requests from the pseudo-nodes, the legitimate nodes will cut off communication. By forcing all nodes to provide location claims, we can rely on the resiliency of our detection algorithm to revoke replicated nodes.

## 8.5 Emergent Properties

Our two primary algorithms, Randomized Multicast and Line-Selected Multicast, achieve distributed detection of global events, while imposing low overheads and providing high resiliency. Their strong properties are a result of their emergent nature. Emergent algorithms utilize the collective efforts of multiple sensor nodes to provide capabilities beyond those of any individual node. For example, after a network's initial deployment, a topology emerges as nodes exchange neighbor information and establish a routing infrastructure. Since emergent algorithms operate in a distributed fashion, they tend to be highly robust in the face of individual node failures, and they avoid the problems inherent in centralized solutions. These properties make them ideal for security applications, particularly in a setting in which we cannot predict the number or identity of the sensors that will be subverted by an adversary. To the best of our knowledge, our protocols represent the first application of emergent algorithms to the problem of security in sensor networks, and we believe that additional research will only continue the trend towards distributed solutions. Furthermore, emergent algorithms apply more generally to security in other classes of networks in which individual nodes are vulnerable to compromise (e.g., peer-to-peer networks and ad-hoc wireless networks), allowing solutions in one class to extend naturally to the others.

## 8.6 Related Work

Eschenauer and Gligor [13] propose centralized node revocation in sensor networks. When the base station detects a misbehaving node, it broadcasts a message to revoke that node. Chan, Perrig, and Song [6] propose a localized mechanism for sensor network node revocation; in their approach, nodes can revoke their neighbors. Neither paper discusses a distributed approach for detecting distributed intrusions.

In broadcast encryption, key revocation has been an important mechanism to recover from compromised keys [2, 3, 15, 16, 21, 26]; however, these approaches do not provide duplicate node detection in sensor networks. Similarly, the results of key revocation in the context of multicast content distribution [35, 37] do not apply here.

The Sybil attack is related to the node replication attack—in the Sybil attack, a malicious node gains an unfair advantage by claiming multiple identities [10]. Douceur presents countermeasures for peer-to-peer networks that involve resource verification (computation, communication, and memory) [10]. Newsome et al. present techniques to defend against the Sybil attack in sensor networks [27]. Their countermeasures include wireless network testing, key space verification, and central node registration. Only their centralized node registration technique (similar to the one we describe in Section 3.1) can detect a node replication attack, but it is brittle in the face of node compromise and has a high overhead, as we show in this paper.

Bawa et al. [1] propose an algorithm for counting the number of members of a peer-to-peer network that is related in spirit to our approach; they propose a random sampling approach that provides an estimate of the network size after $O(\sqrt{n})$ samples are drawn and a duplicate node is sampled (for a network of size $n$), by using the birthday paradox. Our Randomized Multicast also relys on the birthday paradox.

# Chapter 9

# Future Work

In the preceding discussion, we have assumed that the nodes controlled by the adversary continue to follow the protocols described. In our future work, we would like to explore additional mechanisms to ensure that our protocols continue to function even in the face of misbehaving nodes. For example, McCune et al. describe a technique that uses secure implicit sampling to detect nodes that suppress or drop messages [23]. We could also use some of the techniques described in Section 8.2 to periodically sweep the network for replicas, thus preventing the adversary from establishing a significant foothold in the network.

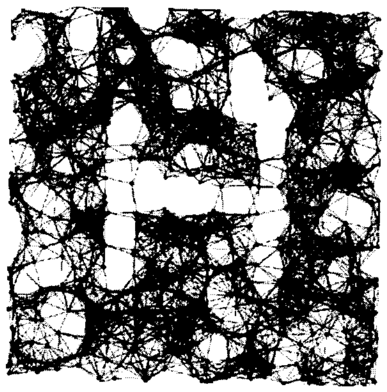
# Chapter 10

# Conclusion

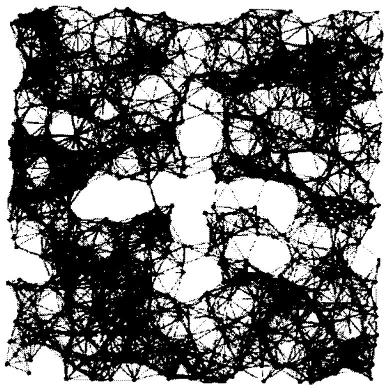
We have discussed various approaches used to detect node replication. In Section 3, we show how centralized approaches place excessive trust in the base station and excessive load on those nodes near it. Local voting schemes are ill equipped to detect distributed node replication. In contrast, we present two schemes that enable distributed detection of distributed events. The final scheme, Line-Selected Multicast, provides excellent resiliency while achieving near optimal communication overhead with only modest memory requirements. Both of our primary schemes illustrate the power of emergent properties in sensor networks. Given the adversary model typically assumed in sensor networks, we argue that the security of such networks will increasingly depend on emergent algorithms. Cost considerations and unattended deployment will always leave individual sensors vulnerable to compromise. Since we cannot predict the exact nature or number of targets the adversary will select, the network must collectively resist, report and revoke compromised nodes in a manner that goes beyond traditional intrusion detection systems. We expect that emergent algorithms will ultimately provide the best defense against these insidious attacks.

# Appendix A

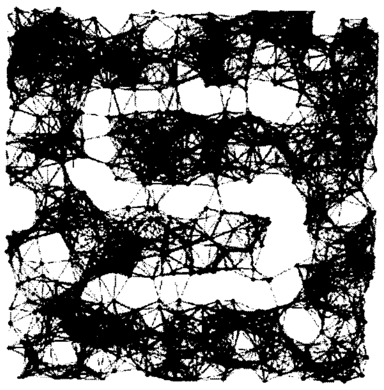
# Network Topologies



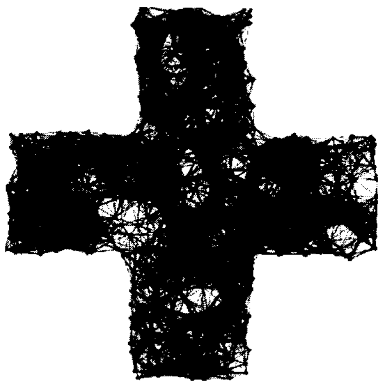
(a) Thin H



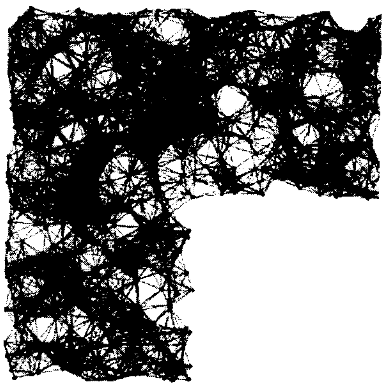
(b) Thin Cross



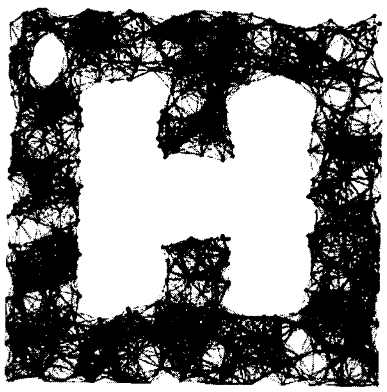
(c) S



(d) Large Cross



(e) L



(f) Large H

Figure A.1: **Assorted Network Topologies** *We tested the Line-Selected Multicast protocol with a variety of network topologies. Samples of each are shown here. Dots represent nodes, and lines represent connections between neighbors.*

# Acknowledgements

The author would like to thank Haowen Chen and Dawn Song for their insightful comments and suggestions. Diana Seymour provided discerning observations and invaluable help editing the paper.

# Bibliography

[1] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani. Estimating aggregates on a peer-to-peer network. Technical report, Stanford University, 2003.

[2] C. Blundo and A. Cresti. Space requirements for broadcast encryption. In *Advances in Cryptology (EUROCRYPT)*, 1995.

[3] C. Blundo, L. Mattos, and D. Stinson. Trade-offs between communication and storage in unconditionally secure schemes for broadcast encryption and interactive key distribution. In *Advances in Cryptology (CRYPTO)*, 1996.

[4] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proceedings of ACM Workshop on Wireless Sensor Networks and Applications*, 2002.

[5] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low-cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, October 2000.

[6] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2003.

[7] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[8] L. Doherty, K. S. J. Pister, and L. E. Ghaoui. Convex position estimation in wireless sensor networks. In *Proceedings of IEEE Infocom*, 2001.

[9] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 1983.

[10] J. R. Douceur. The Sybil attack. In *Proceedings of Workshop on Peer-to-Peer Systems (IPTPS)*, Mar. 2002.

[11] J. Dyer, M. Lindemann, R. Perez, R. Sailer, L. van Doorn, S. W. Smith, and S. Weingart. Building the IBM 4758 Secure Coprocessor. *IEEE Computer*, 2001.

[12] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 2002.

[13] L. Eschenauer and V. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, Nov. 2002.

[14] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Mobile Computing and Networking*, 1999.

[15] A. Fiat and M. Naor. Broadcast encryption. In *Advances in Cryptology (CRYPTO)*, 1994.

[16] J. Garay, J. Staddon, and A. Wool. Long-lived broadcast encryption. In *Advances in Cryptology (CRYPTO)*, 2000.

[17] V. D. Gligor. Security of emergent properties in ad-hoc networks. In *Proceedings of International Workshop on Security Protocols*, Apr. 2004.

[18] A. Hu and S. D. Servetto. Asymptotically optimal time synchronization in dense sensor networks. In *Proceedings of ACM International Conference on Wireless Sensor Networks and Applications*, 2003.

[19] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of Conference on Mobile Computing and Networking (MobiCom)*, Aug. 2000.

[20] D. Liu and P. Ning. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, Feb. 2003.

[21] M. Luby and J. Staddon. Combinatorial bounds for broadcast encryption. In *Advances in Cryptology (EUROCRYPT)*, 1998.

[22] D. Malan, M. Welsh, and M. Smith. A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In *Proceedings of IEEE Conference on Sensor and Ad hoc Communications and Networks (SECON)*, Oct. 2004.

[23] J. M. McCune, E. Shi, A. Perrig, and M. K. Reiter. Detection of denial-of-message attacks on sensor network broadcasts. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2005.

[24] R. Merkle. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Apr. 1980.

[25] R. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology (CRYPTO)*, 1988.

[26] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology (CRYPTO)*, 2001.

[27] J. Newsome, E. Shi, D. Song, and A. Perrig. The Sybil attack in sensor networks: Analysis and defenses. In *Proceedings of IEEE Conference on Information Processing in Sensor Networks (IPSN)*, Apr. 2004.

[28] J. Newsome and D. Song. GEM: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2003.

[29] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *ACM Conference on Mobile Computing and Networks (MobiCom)*, July 2001.

[30] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A geographic hash table for data-centric storage. In *Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Sept. 2002.

[31] P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, Nov. 1999.

[32] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla. Swatt: Software-based attestation for embedded devices. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2004.

[33] S. W. Smith and S. Weingart. Building a high-performance, programmable secure coprocessor. *Computer Networks*, Apr. 1999. Special Issue on Computer Network Security.

[34] H. Solomon. *Geometric Probability*. Society for Industrial and Applied Mathematics (SIAM), 1978.

[35] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. Internet Request for Comment RFC 2627, Internet Engineering Task Force, June 1999.

[36] S. Weingart. Physical security devices for computer subsystems: A survey of attacks and defenses. In *Cryptographic Hardware and Embedded Systems (CHES)*, Aug. 2000.

[37] C. Wong, M. Gouda, and S. Lam. Secure group communications using key graphs. In *Proceedings of ACM SIGCOMM*, 1998.