ANONYMITY AND HOSTILE NODE IDENTIFICATION IN WIRELESS SENSOR NETWORKS

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Phillip Steven Reindl

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

February 2010

Fargo, North Dakota

North Dakota State University
Graduate School

Title

ANONYMITY AND HOSTILE NODE IDENTIFICATION

IN SENSOR NETWORK

By

PHILLIP STEVEN REINDL

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

ABSTRACT

Reindl, Phillip Steven, M.S., Department of Computer Science, College of Science and Mathematics, North Dakota State University, February 2010. Anonymity and Hostile Node Identification in Wireless Sensor Networks. Major Professor: Dr. Kendall Nygard. Co-Advisor: Dr. Xiaojiang Du.

In many secure wireless network attack scenarios, the source of a data packet is as sensitive as the data it contains. Existing work to provide source anonymity in wireless sensor networks (WSN) are not frugal in terms of transmission overhead. We present a set of schemes to provide secure source anonymity. As the state of the art in WSN advances, researchers increasingly look to heterogeneous network topologies. We leverage high powered cluster head nodes to further reduce transmission overhead and provide excellent scalability. A significant threat to WSN is the insider attack due to the ease of tampering with low-cost sensors. Should a node become compromised and start making malicious collisions, it is desirable to identify the corrupt node and revoke its keys. We present schemes to identify the source of an arbitrary transmission in a reliable and distributed fashion.

ACKNOWLEDGMENTS

I would like to thank my wife, Katie, and my two wonderful sons, Ivan and Jonah, for giving me the freedom and support to pursue this master's degree. I would also like to thank the National Aeronautics and Space Administration's Space Grant program for funding my research in 2007-2009. I would also like to thank Dr. Du, without whose encouragement I would never have entered the graduate program. Finally, I would like to thank Dr. Nygard, whose eleventh hour support has been invaluable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure | Page

CHAPTER 1. INTRODUCTION

Achieving high security efficiently is a fundamental challenge in wireless sensor networks (WSN) today. Unlike wired networks, WSN use a noisy channel for communication that is easily tapped by an adversary. Eavesdropping is a critical concern since it is such a trivial attack to launch. In many asset monitoring applications, the source of an event data packet is as sensitive as the contents of that packet. Our work in Chapter 2 attempts to find an efficient answer to the question "How can the source of a data packet be hidden in a wireless network?" Making the issue even more challenging is the fact that nodes in WSN are often deployed in unattended, hostile environments. Energy is a precious commodity that cannot be squandered. Both computation and transmission are costly in the wireless economy [1]. Rather than all nodes simply sending dummy packets periodically, short control packets are used to coordinate the more costly dummy packets. By utilizing control packets, we are able to demonstrate substantial energy savings while maintaining source anonymity.

The vision of WSN is to be able to deploy thousands of nodes by random scattering. A consequence of this is that each node must be very cheap. We therefore must assume that the bulk of the nodes in the network are not equipped with any tamper resistant hardware, and so can be easily compromised. A compromised node will have the full contents of its memory revealed, and may behave arbitrarily [2]. Under this assumption, we cannot afford to trust any given common node to behave correctly. Such a compromised node might send false messages in an attempt to waste system resources or otherwise disrupt communication. Chapter 3 presents new distributed schemes to find the source of any given transmission.

Zhu, *et al.* [3] suggested that there is a minimum time for an adversary to compromise a node, T_{min}, and the estimated time for the network to self-organize, T_{est}. If $T_{est} < T_{min}$ then it is fair to assume that the network is secure during deployment. Thus, during network deployment we assume a very short time frame where the network is secure. All nodes behave correctly because the adversary hasn't had time to compromise any of them. During this time, fingerprints are generated for all nodes in the network. Subsequent messages are compared against these fingerprints to find the source.

1.1. Network Model

In our model, we consider three classes of wireless node. Low-powered L-nodes, much more capable H-nodes, and a single Base Station. Much of the current literature assumes only L-nodes and a single Base Station. This is the case in [4] and [5], which are the primary works we build on in Chapter 2. The L-nodes are underpowered nodes with little processing or storage capabilities. The Micaz mote is an appropriate reference platform [6]. L-nodes are starved for processing capability and battery power. They also lack tamper-resistant hardware, and can not be trusted.

Due to issues of scalability and security, current research is considering heterogeneous networks, which add H-nodes acting as cluster heads. The H-nodes are relatively powerful, with storage and computation roughly equivalent to a PDA. The radio of the H-node is able to reach neighboring H-nodes, and thus can broadcast directly to all L-nodes in its cluster. Furthermore, H-nodes are generally assumed to be equipped with tamper-resistant hardware, and can thus be assumed to be trustworthy.

The simple assumption that the cluster head (H-node) is trustworthy and can simultaneously broadcast to all L-nodes in the cluster has profound impact. Since H-nodes are able to reach all L-nodes in the cluster simultaneously, clock synchronization within the cluster becomes trivial. H-nodes having tamper resistant hardware provide excellent aggregation points.

In-network aggregation is an important technique to reduce total network traffic, conserving precious battery power [5], [7]. H-nodes are powerful enough to perform statistical tests or otherwise remove spurious data. H-nodes are also able to manage cryptographic keys for the L-nodes in the cluster. Finally, H-nodes are able to form a second-tier backbone network [1]. Not only is the traffic reduced by aggregation, it is reduced by longer transmission ranges – H-nodes cover the same geographic distance with fewer hops.

The Base Station is considered to be a laptop-class computer with vast computation, storage, and energy capabilities.

1.2. <u>Threat Model</u>

We are concerned with both a global, passive observer, and an active insider.

A global, passive observer is reasonable because an adversary can deploy a low-cost network in the same area as the target network. The attacker's network can simply record radio traffic and relay the results to a central location for analysis. For this reason, we must encrypt all sensitive data. It is also important to anticipate traffic analysis. Which nodes are sending data might be enough information for the adversary to accomplish his goals.

An active insider is a reasonable threat because of the nature of the L-nodes. Since they are not fitted with tamper-resistant hardware, they can be easily compromised, and may be under the control of the adversary. Thus, nodes in the network should not have access to any more information than is necessary to accomplish their job. Furthermore, claims made by individual L-nodes should not be trusted.

All schemes running on WSN must be examined in the context of these threats. Careful application design will consider what aspects of the specification can be used against the network. Can a single node strongly influence the behavior of a significant number of peer nodes? What measures can be taken to discover an attacker?

1.2.1. Selective Forwarding

In the Selective Forwarding attack, a node will drop packets in an effort to disrupt communications. An unsophisticated attacker will simply drop packets at random, at a rate that will not raise suspicion. If the node is able to read the contents of the messages it routes, a more sophisticated attack can be carried out where only traffic from a few specific sources are dropped. In this case, routing information can be the weak point, even if the payload is unknown.

The basic Selective Forwarding attack is carried out by a single corrupt node, and applies only to routes that it is on. This specific attack is outside the scope of the work presented. Chapter 3 provides techniques for detecting the source of spurious transmissions such as would be generated by an adversary in the Malicious Collision attack (below.)

1.2.2. Malicious Collision

Malicious Collision is a significant threat against wireless sensor networks. Law, *et al.*, [8] address the issue with the attack model of an outsider deploying a jamming network in the same area as the target network. They found that an attacker can effectively silence the network very efficiently if the traffic patterns are predictable. We consider the stronger insider threat. Corrupt nodes carrying out an insider attack is a stronger threat due to their knowledge of the traffic, however we do not expect more than a few nodes to be compromised.

The goal of malicious collision, just as other selective forwarding attacks is to prevent data from a specific source or event from reaching the cluster head or Base Station.

1.2.3. Malicious Collision with Collusion

Two corrupt nodes can collude to effect the malicious collision attack. This is done by having the nodes coordinate. In this case, only one of the nodes in on the route, and the other disrupts the transmissions to the downstream neighbor. Since the collisions are coordinated, the MAC layer handshaking can be disrupted in addition to the data packet.

For example: Using out of band communication, *v* will be able to alert *x* that it is about to transmit a packet that should be disrupted. In this case, *v* and *x* collude to disrupt the packet and the MAC layer handshaking preceding the packet. This is the key difference between this scenario and the simple malicious collision attack. In this case, the transmission will appear to have been completed correctly to upstream neighbor *u*, and downstream neighbor *w* will be unaware that a packet was just lost. *w* will not be able to tell the difference between the collision it sees and a "collision" it sees due to the hidden terminal problem. E.g., Two neighbors of *w* might be sending packets independent of each other, and *w* will see a collision even if neither of the neighbors transmitting do.

1.3. <u>Security Properties</u>

1.3.1. Source Anonymity

For some applications, it is important to protect not only the data of the event, but also the source of the event. A sophisticated selective forwarding or traffic analysis attacker can use the source of the information to his advantage. For example, if the sensor network detects specific events, the fact that data traffic is being generated at a given source means that events of that type are occurring near the source. Conversely, an attacker might want to darken a specific region of the network. If they are able to disrupt traffic only from those specific sources, detection of the attack is more difficult.

Source anonymity precludes static routes. If an intermediate node is going to route a packet correctly, it has to know which route to use. Each route is associated with a single source. For this reason, to reduce the threat, routes should be identified by a single use pseudonym. Pseudonyms can be generated using a one-way key chain function as used in μ-tesla [9], [10]. For each message a node sends, a different key is used in the key chain. Without the generating key, the next key is not predictable.

Under the threat of a global observer, route pseudonyms are not strong enough since the observer will be able to observe the original transmission regardless of the pseudonym used.

1.3.2. Event Unobservability

Event unobservability means to hide the fact that an event has occurred to an external observer. An adversary might be able to learn useful information simply by observing network traffic and seeing that something has been detected. Even if the source is not revealed, this might represent a security breach in some scenarios.

A common technique to provide Event Unobservability is using probabilistic dummy traffic mix (pdtm) to thwart traffic analysis. The pdtm algorithm will determine the delay between sending packets, and generate dummy messages if necessary. For protection against insiders, dummy traffic should be obscured to the L-nodes, and thus must be routed to H, at great expense. The problem of event unobservability is addressed by [4], [5], [11]. The baseline scheme in Chapter 2 has this property, but we relax the requirement to achieve greater transmission efficiency, while still providing Source Anonymity in the presence of a global observer.

1.4. Work Overview

In this work, two major topics are considered. Chapter 2 addresses the issue of event source anonymity. In some security contexts, the observation of network activity represents a security compromise. We demonstrate a technique for providing a lightweight security mechanism for protecting the source of event data in the presence of a passive, global observer.

In Chapter 3 we discuss a technique for identifying the source of an arbitrary wireless transmission from among the network deployment. Our security model assumes a network consisting mainly of untrusted Low-Powered nodes. In the event that a node becomes compromised it can generate arbitrary messages, causing mayhem. It is therefore crucial to the proper long-term functioning of the network to be able to identify misbehaving nodes.

CHAPTER 2. LIGHTWEIGHT SOURCE ANONYMITY

2.1. Introduction

For many applications of Wireless Sensor Networks (WSN), the source of the event data is as sensitive as the event data itself. In the resource constrained WSN, providing source anonymity is a challenging problem. A traditional approach for event hiding in WSN is for nodes to periodically generate dummy traffic even if they have no event data to send. The observer will have no way to differentiate the data packets from the dummy packets. This simplistic approach generates an enormous amount of traffic which must be routed to the Base Station. The volume of traffic increases directly with the number of sensors in the network, quickly becoming a prohibitive expense to network operation. We will detail a lightweight, scalable solution.

We expound on the previous work [4] by adding *control packets* which are short messages used to coordinate the transmission of longer, less frequent *data packets*. Source anonymity is maintained. An observer will only know that an event has occurred, but will be unable to determine what the event was or where it happened. Note that this is a weaker property than previous work, which also obscured the occurrence of any event. For many applications, such as the hunter/prey model presented below, this is acceptable. (E.g., the hunter already knows there are pandas in the area, and gains nothing by knowing that one was observed somewhere in the deployment area.)

Further, by leveraging high-powered nodes, the Two-Tier scheme demonstrates much greater scalability than the traditional homogeneous networks are able to achieve. In a homogeneous network of low-powered nodes (L-nodes), the overhead of forwarding packets becomes prohibitive. Particularly, the nodes nearest the Base Station carry a substantially higher traffic load than distant nodes [12]. This disparity causes the nodes closest to the Base Station to fail prematurely. When all links to the Base Station have failed, the usefulness of the network is over.

In-network aggregation is the clear solution to reduce traffic. Great care must be taken to ensure that the aggregation is done securely. The aggregation points must be able to differentiate the traffic (dummy or data) and thus make attractive targets for compromise. Schemes that rely exclusively on L-nodes such as [5] have a particularly difficult time proving security since the L-nodes are generally assumed to be easily compromised [2].

2.2. Related Work

In the previous work [4], source anonymity is maintained by sending dummy traffic with probabilistic delay between messages. A passive, global observer will be unable to differentiate the dummy traffic from event messages. Rather than using a constant delay between messages, the FitProbRate scheme varies the inter-message delay to be the minimum fitting some random distribution (e.g., exponential.) [5] expands on this work by introducing proxy nodes which filter the dummy traffic, reducing the overall network energy dissipation. This method is still expensive, as the dummy messages are the same length as the event messages. Furthermore, the proxy nodes represent failure points that can disrupt the network over a significant area.

Ahn, Bortz, and Hopper [13] introduced the concepts of k-anonymity. They provide a scheme giving both sender and receiver k-anonymity. In a network of n nodes, full anonymity is achieved when k = n. By limiting the anonymity to a subset of n, anonymity acceptable for many applications is achieved at much lower cost. Their scheme relies upon public key encryption, and is thus unsuitable for most wireless sensor networks [3].

The approach to efficient source anonymity for sensor networks taken by [14], [15] is to send the data packets on a random walk before routing to the base station. The random walk routes the data to an intermediate destination before final routing to the base station. This layer of misdirection can add a degree of security, but is not robust against a global observer.

ANODR [16] provides route anonymity and location privacy for ad-hoc networks. Route pseudonyms are employed which require a costly set-up phase. The burdens of trap-door encryption and route setup are difficult to justify in the energy-constrained wireless sensor network (WSN). Furthermore, the mixing techniques employed by ANODR require constant traffic with varied sources and destinations. The WSN has relatively light traffic and a fixed destination for all data flows.

2.3. Assumptions

2.3.1. Network Model

The baseline scheme, FitProbRate, and our homogeneous scheme (HSA) assume a homogeneous sensor network consisting of low powered L-nodes. All radio links are bidirectional. A MAC layer protocol is in place to provide reliable communications between neighboring nodes.

The two-tier scheme (TTSA) assumes a heterogeneous network, consisting of many clusters. Each cluster has a single high-powered H-node acting as cluster head over many L-nodes. L-node to L-node and and H-node to H-node communications are bidirectional, but H-node to L-node communication is unidirectional. Only L-nodes close to the cluster head are able to send messages to it. Again, a MAC layer protocol is assumed to synchronize transmissions and give reliable communications.

In all cases, transmissions are encrypted such that an outside observer (the hunter – See section 2.3.2.) cannot derive the message contents. We also assume that clocks of all nodes in the network are synchronized. Finally, we assume that all data is routed to an immobile base station.

2.3.2. Attack Model

The scenario posed in [4], [5] is a sensor network monitoring endangered animals such as giant pandas. The attacker is a hunter who has placed a sensor network in the same area that can monitor radio transmissions. We are thus concerned with a global, passive observer. In this scenario, the hunter's goal is to use traffic analysis to find the source of event data, which is where the panda is located.

Inexpensive low power sensor nodes (L-nodes) are unlikely to be fitted with tamper-resistant hardware, and are generally assumed to be easily compromised [2]. We must therefore be concerned with an insider threat when considering the L-nodes. We will assume that there is an initial setup period where all nodes are trusted [3], during which time cryptographic keys are shared between neighboring nodes. After the secure period, we assume that some minority of the network is compromised and may behave arbitrarily.

The high-powered cluster head H-nodes (used only in two-tier scheme) are more costly and are assumed to be fitted with tamper-resistant hardware [17]. No H-nodes are compromised.

2.3.3. Security Framework

L-nodes are incapable of utilizing asymmetric key cryptography due to their limited resources [3]. Each L-node shares a symmetric key with the BS or cluster head. This key is used to encrypt data packets sent from the node to the BS or cluster head. It is also used when computing the SourceID, below.

Each L-node has a broadcast key known by its neighbors. This key is used for local broadcasts to encrypt data from an outsider. Finally, each L-node shares a pair-wise key with each of its neighbors.

Each H-node has a broadcast key known by its neighbors, a pairwise key for each neighbor and a pairwise key shared with the BS. In the Two-Tier scheme, H-nodes act as cluster heads, requiring a pairwise key shared with each L-node in the cluster.

This is essentially the keying scheme used in [3], extended to apply to Two-Tier networks. We denote a key shared between two nodes *i, j* as follows: $K_{i,j} = K_{j,i}$. The broadcast key of node *i* is denoted as K_{Bi}.

2.3.4. MAC Layer

To our knowledge, an efficient MAC layer protocol for heterogeneous sensor networks has yet to be defined in literature. When designing a heterogeneous sensor network, an issue that needs to be addressed is assuring that the cluster head will be able to be heard by all the nodes in the cluster without collisions. 802.11 type rts/cts is ineffective because the distant L-nodes won't be able to negotiate with the H-node due to their limited transmission power. TDMA solves the negotiation problem, but is inefficient during normal network activity [18].

We propose an 802.11/TDMA hybrid MAC layer protocol. Time will be divided into two asymmetric slots, one for the cluster head to broadcast to the cluster (minority slot), the other, longer time slot (majority slot) for the L-nodes to communicate with each other.

If the timing of these slots is static, a compromised node will be able to interfere with the cluster head acknowledgments since the timing could be predicted. For this reason, we will use a variable time based on a pseudorandom number seed given by the cluster head. When nodes are revoked from the network, a new seed will be issued so that the revoked nodes won't be able to predict the timing.

Before sending a broadcast message, all nodes sense the channel to minimize collisions.

2.4. Effective Schemes for Source Anonymity

2.4.1. FitProbRate - FPR

Our schemes are compared with the FitProbRate scheme proposed in [4]. FPR assumes a homogeneous network and an outside attacker performing traffic analysis. End-to-End encryption is used so intermediate L-nodes are unable to determine whether a given packet is dummy traffic or data traffic.

The main contribution is the statistically strong minimal delay for event data. All nodes send data traffic with a random delay with mean λ. When a node has event data to send, it finds the minimum delay that still fits the probability distribution with a given confidence threshold. Shao, *et al.* found that the FitProbRate [4] scheme reduced latency from 10.87s for the constant rate scheme to under 1s. No consideration was given to the traffic overhead required, which is what we seek to improve in our schemes. We use FPR as a baseline scheme for comparison.

The procedure to find the minimum delay given in [4] is for each node to track the history of delays and select a new delay that is as small as possible yet fitting the probability distribution with the desired statistical significance. Procedure MinDelay (Figure 2) is used to find the minimum delay. Care must be taken to recover the mean λ before the usual distribution function can be used again. Procedure Recover (Figure 3) is used to recover the proper mean. Both procedures use the Anderson-Darling (A-D) Test [19] to verify the fit. The A-D test (Figure 1) uses test statistic A^2 (1).

$$A^2 = -n - \sum_{i=1}^{N} \frac{2i-1}{n} \left[\log(F(X_i)) + \log(1 - F(X_{n+1-i}))\right] \tag{1}$$

- n is the sample size
- X_i is the ith delay
- F is the Cumulative Density Function (CDF) for the probability distribution being used. We assume the Exponential probability function in this work, the CDF is given in (2), below.

$$F(x;\lambda) = \begin{cases} 1 - e^{-\lambda x} & if\,(x \geq 0) \\ 0 & if\,(x < 0) \end{cases} \tag{2}$$

2.4.1.1. Estimating Overhead for FPR

There are two main forms of overhead that we are concerned with minimizing: Transmission overhead and Latency. Transmission overhead is the amount of traffic that is required by the scheme that wouldn't be required otherwise. Latency is the sum of all delays imposed by the scheme over simple routing.

We will use the following notations to form the estimates:

Algorithm Anderson-Darling [4]

Input: a sequence of data $\{x_i, 1 \le i \le n\}$, critical value c

Output: TRUE, if $\{x_i, 1 \le i \le n\}$ follows an exponential distribution, FALSE otherwise

```
1.    sort $x_i$ into ascending order
2.    calculate $A^2$ for sorted data
3.    if ( $A^2 < c$ ) then
4.        return TRUE
5.    else
6.        return FALSE
7.    endif
```

Figure 1. Algorithm to compute the Anderson-Darling goodness of fit test.

Algorithm MinDelay [4]

Input: a sequence of delays $\{x_i, 1 \le i \le n\}$, delay mean λ

Output: a minimal delay fitting the desired exponential probability distribution

```
1.    $x := 0$
2.    while ( Anderson-Darling($\{x, x_2, x_3, ..., x_n\}$) = FALSE )
3.        if ($x > \lambda$) then
4.            $x := 0$
5.        endif
6.        $x := x$ + rand(0, $\lambda/4$)
7.    loop
8.    return $x$
```

Figure 2. Algorithm to find minimum delay fitting the exponential probability distribution with the given delay history.

Algorithm RecoverMean [4]

Input: a sequence of delays $\{x_i, 1 \le i \le n\}$, delay mean λ

Output: a proper delay to restore the mean

```
1.    $sum := \sum_{i=2}^{n} x_i$
2.    $dx := \lambda - sum/(n-1)$
3.    $target := n(\lambda + dx) - sum$
4.    $range$ := exponential($\lambda$)
5.    do
6.        $x$ := rand($target$, $range$)
7.    while ( Anderson-Darling($\{x, x_2, x_3, ..., x_n\}$) = FALSE )
8.    return x
```

Figure 3. Algorithm to find a delay to recover the proper mean of the exponential probability distribution with the given delay history.

- There are n nodes in the network.
- Nodes are an average of d hops from the BS.
- Every node sends a data packet every λ seconds, independent of whether it has data to send.
- Network lifetime in seconds is Φ.
- Packets are δ bytes each.
- The number of events during network lifetime is Ω.

Total traffic overhead generated in FPR is $n\frac{\Phi}{\lambda}d\delta-\Omega d\delta$ bytes. Every node sends a data packet every λ seconds, independent of whether it has data to send. The packets must be routed an average of d hops to reach the BS. There are Ω events during the network lifetime, each of which generates a single packet which must be routed to the BS. These Ω packets are the cargo the scheme is protecting, and are deducted from the total overhead.

Average latency is $\frac{\lambda}{10}$ seconds. When a node has data to send, the data is sent directly to the BS with a delay given by MinDelay.

2.4.2. Scheme for Homogeneous WSN - HSA

The Homogeneous sensor network consists of many L-nodes which route data to a central base station. In order to provide source anonymity, dummy packets are sent by nodes that do not have data to send. The dummy packets are very costly, so smaller control packets are used to coordinate their transmission throughout the network.

Control packets are very short, containing only the node id, a bit signifying whether there is data to send, and the time to send. 5 bytes is adequate for most applications, allowing for 32k unique node ids. The TimeToSend field will contain a random value if there is no data to send.

PrevHop	K_{Bj}\{SourceID | DataToSend | TimeToSend\}

Figure 4. Control packet format.

The Control packet has four fields: PrevHop, SourceID, DataToSend, TimeToSend.

PrevHop: 15 bit field containing the ID of the L-node j making the transmission. This is needed so that neighboring L-nodes are able to use the correct broadcast key K_{Bj} to decrypt the remainder of the packet.

SourceID: 15 bit field containing the pseudo-ID of the node that detected the physical event, and sent the first '*yes*' control packet. The pseudo-ID is computed using a one-way hash function $f(ID_S, i_S, K_{S,BS})$

ID_S	The true ID of the source node S
i_S	A monotonically increasing sequence number for the '*yes*' control packets initially sent by node S.
$K_{S,BS}$	Symmetric key shared by source node S and the Base Station.

Since f depends on the key $K_{S,BS}$ known only by the source node S and base station, only the base station is able to determine ID_S. The base station will track how many '*yes*' control packets were sent by each node, and using these values compute the next several pseudo-IDs that will be used by each node.

DataToSend: 1 bit field. '1' means this is a '*yes*' control packet, '0' means this is a '*no*' control packet.

- *Ready* nodes will generate '*yes*' control packets, with the TimeToSend field set to their TTS.

- *Idle* nodes will set their TTS to the TimeToSend field of a '*yes*' packet they receive, and become *ready*.

TimeToSend: 24 bit field containing the expected time μ to send data packets. In order to avoid collisions that would occur if all nodes in the network broadcast simultaneously, each node broadcasts its data packet at a time given by the Gaussian distribution with mean μ.

K_{Bj}: The broadcast key of node *j*, used to encrypt the payload of the packet.

If the TimeToSend field specifies the time in seconds, a network lifetime of 194 days is allowed before the clocks overflow. If L-nodes are not re-keyed at that time, a patient attacker could conduct a replay attack. The goal of such an attack is to drain system resources by sending false '*yes*' control packets.

Data packets are much longer than control packets, containing the actual event data. We will use 30 byte data packets for calculations, as this is a standard data packet in existing systems such as TinyOS [20]. The contents of the data packet is highly application dependent, so we make no assumptions other than the size of the data packet.

As in previous work [4], nodes transmit traffic continuously following a random probability distribution. We will assume the exponential distribution since it only has one parameter, the mean λ. Most of the time, this traffic will consist only of control packets. An *idle* node will send '*no*' control packets with a delay fitting the probability distribution, a *ready* node will send '*yes*' control packets with the minimum delay that follows the distribution. A node moves to the ready state when one of two conditions have been met:

1) It has data to send.

2) It has received a '*yes*' control packet.

Since *ready* nodes continuously send '*yes*' control packets, if any node has data to send, all nodes in the network will eventually move to the ready state.

All the clocks in the sensor nodes are synchronized. Each node has an estimate, T_P, for how long it takes a message it sends to propagate through the network. An *idle* node with event data to send, will become a *ready* node, with Time To Send (TTS) set to the current time + T_P. An *idle* node that receives a '*yes*' control packet will use the TTS from the incoming '*yes*' packet.

Shao, *et al.* found that the FitProbRate [4] scheme reduced latency from 10.87s for the constant rate scheme to under 1s. Based on their result, we use an accelerated transmission delay of $\lambda/10$ in our simulations. Each node will set $T_P = \frac{\lambda}{10}H$, where H is how many hops the node is from the far edge of the network.

If T_P is overestimated, then nodes will wait longer than is necessary for the '*yes*' control packets to propagate through the network, causing increased latency. On the other hand, if T_P is underestimated, then the '*yes*' control packets will not have time to propagate through the network, and some nodes will not send dummy traffic. The effect is to give the adversary clues to the location of the event.

When the current time matches the TTS, all ready nodes (the entire network if T_P is estimated correctly) send a data packet to the base station. Nodes with no data to send will generate a dummy data packet. All sensors reset to the *idle* state, and the process repeats.

A concrete description is given in Figure 5, below.

```
Algorithm Homogeneous Source Anonymity
Input: Distance H the node is from the far edge of the network (Hops)
       Mean delay between control packets $\lambda$
1.   $Tp := \frac{\lambda}{10} H$
2.   $\varphi := \{\}$
3.   hasData := FALSE
4.   state := IDLE

IDLE state
→ 'no' Control Packet received
1.   state := IDLE

→ 'yes' Control Packet received
1.   ctrlDelay := MinDelay($\varphi$, $\lambda$)
2.   set ControlTimer
3.   dataDelay := packet[tts] - CurrentTime
4.   set DataTimer
5.   state := READY

→ ControlTimer fired
1.   $\varphi := \varphi \cup \{ctrlDelay\}$
2.   send 'no' Control Packet
3.   ctrlDelay := RecoverMean($\varphi$, $\lambda$)
4.   set ControlTimer
5.   state := IDLE

→ Sensor Event
1.   set hasData := TRUE
2.   ctrlDelay := MinDelay($\varphi$, $\lambda$)
3.   set ControlTimer
4.   dataDelay := Tp
5.   set DataTimer
6.   state := READY

READY state
→ 'no' Control Packet received
→ 'yes' Control Packet received
1.   state := READY

→ ControlTimer fired
1.   $\varphi := \varphi \cup \{ctrlDelay\}$
2.   send 'yes' Control Packet
3.   ctrlDelay := RecoverMean($\varphi$, $\lambda$)
4.   set ControlTimer
5.   state := READY

→ DataTimer fired
1.   if hasData = TRUE then
2.        send 'Event' Data Packet
3.   else
4.        send 'Dummy' Data Packet
5.   endif
6.   hasData := FALSE
7.   state := IDLE

→ Sensor Event
1.   hasData := true
2.   state := READY
```

Figure 5. Algorithm describing behavior of nodes in scheme HSA. Procedure MinDelay is given in Figure 2. φ maintains the history of Control Packet delays.

2.4.2.1. Observations

Since the '*yes*' control packets are propagated by continuous flooding until the state changes back to *idle*, it is difficult for corrupt nodes to prevent the rest of the nodes from becoming *ready*.

Since data packets are already differentiated from the control packets, we do not need to delay the data packets. Consequently, the bulk of the latency is a result of the control packet propagation.

Individual control packets can be verified by the BS. The BS will track i_S for all L-nodes in the network. It is then a simple calculation to find the next several pseudo-IDs that will be used by each L-node when they detect an event. When a '*yes*' control packet arrives, the BS will compare the SourceID of the control packet with the set of expected pseudo-IDs. If a match is found, the corresponding node is the source of the control packet. If no match is found, the control packet is fraudulent.

Algorithm Verify

```
1.  foreach L-node n
2.      for seq := n.last to n.last + ω
3.          if f(n.ID, seq, n.key) = pseudo-ID then
4.              n.last := seq
5.              return PASS
6.          endif
7.      endfor
8.  endforeach
9.  return FAIL
```

Figure 6. Algorithm to verify whether the given pseudo-ID is valid.

A corrupt node is not expected to behave correctly. A weakness of HSA is that it is subject to a resource depletion attack: A corrupt node can send false '*yes*' control packets. Since they aren't verified by the L-nodes before sending a data packet, a single corrupt node can force the entire network to send spurious dummy data packets, wasting resources. For this reason, L-nodes track the SourceID given in '*yes*' control packets, and the upstream neighbor who first sent it.

In the event that the L-nodes send data traffic consisting only of dummy packets, the base station can query the network to determine which node was the source of the '*yes*'

Control packet. L-nodes will respond to the query with a message containing their id, time of receipt of the first control packet with the given SourceID, and the upstream neighbor. The base station will use the results of this query to build a tree, the root is the alleged source of the packet in question. This is possible because the SourceID doesn't change from hop to hop and the clocks are synchronized among the L-nodes. An example tree built using this approach is given in Figure 7.

A corrupt node could impersonate a neighbor in an attempt to avoid detection. In order to find the true source of a fraudulent control packet, an RSSI based wireless fingerprinting scheme can be used such as those discussed in Chapter 3. To enable fingerprinting, L-nodes will record the SourceID, RSSI, and timestamp of '*yes*' control packets. When the source is identified, all keys of that node are revoked.

2.4.2.2. Estimating Overhead for HSA

There are two main forms of overhead that we are concerned with minimizing: Transmission overhead and Latency. Transmission overhead is the amount of traffic that is required by the scheme that wouldn't be required otherwise. Latency is the sum of all delays imposed by the scheme over simple routing.

We will use the following notations to form the estimates:

- There are n nodes in the network.
- Nodes are an average of d hops from the BS.
- Nodes are an average of h hops from the far edge of the network.
- Every node sends a control packet every λ seconds, independent of whether it has data to send.

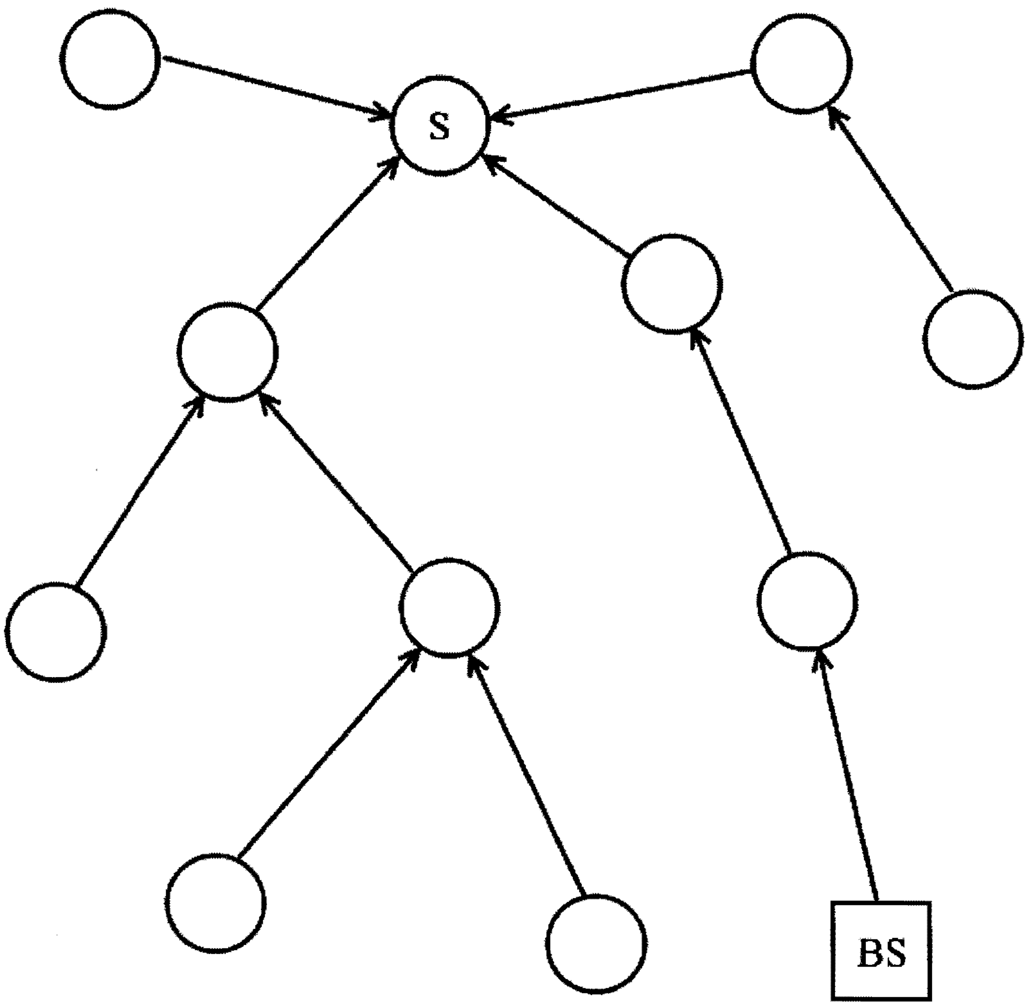

Figure 7. Upstream neighbor tree example. Source of packet in question is node 'S'. 'BS' denotes base station.

- Network lifetime in seconds is Φ.
- Data packets are δ bytes each.
- Control packets are α bytes each.
- The number of events during network lifetime is Ω.

Total traffic overhead generated in HSA is $n\Omega d\delta + n\frac{\Phi}{\lambda}\alpha - \Omega d\delta$ bytes. Every node sends a data packet for every event that occurs. The second term is the traffic due to control packets. Every node sends a control packet every λ seconds on average. There are Ω events during the network lifetime, each of which generates a single data packet which must be routed to the BS. These Ω packets are the cargo the scheme is protecting, and are deducted from the total overhead.

Average latency is $\frac{\lambda}{10}h$ seconds. This is the time it takes 'yes' Control Packets to propagate through the network. When the data timer fires, all nodes send their data packet with no further delay.

2.4.3. Scheme for Two-Tier Heterogeneous WSN - TTSA

The two-tier network is composed of two similar schemes working in concert that make different assumptions. The Intra-cluster functionality (Section 2.4.3.1) concerns the behavior within a single cluster, and considers many L-nodes and a single H-node which is able to make a transmission to all nodes in the cluster simultaneously. The Inter-cluster functionality (Section 2.4.3.2) concerns the behavior of the H-nodes in relation to each other and the BS.

Events are detected by the L-nodes. '*yes*' control packets propagate through the original cluster, and then through the H-nodes. The H-nodes will make a data request from their clusters, prompting all L-nodes in the network to generate a data packet. The data packets are routed to the cluster heads, where they are aggregated and a summary for each cluster is routed to the base station.

2.4.3.1. Intra-cluster Functionality

Within a cluster, the network consists of many L-nodes and a single H-node which is able to transmit to all L-nodes simultaneously. A simple change will dramatically improve performance relative to HSA. By far, the greatest factor contributing to latency in HSA is the control packet propagation delay T_P. Since H is able to reach all nodes simultaneously, if H sends a '*yes*' control packet, all nodes can immediately move to the *ready* state and send data packets. With this observation, it is easy to see that the TTS field is not needed. There are two consequences of removing the TTS field.

1) The control packet is further reduced in size, reducing traffic overhead significantly.
2) Latency will be reduced to its optimal. Nodes transmit data packets only in response to H sending a '*yes*' packet, so there is no danger of L-nodes over- or under-estimating T_P.

PrevHop	K_{Bj}\{SourceID | DataToSend\}

Figure 8. L-Control Packet Format.

The reduced Control packets are referred to as L-Control packets. There are three fields, PrevHop, SourceID and DataToSend. The fields are the same as those in HSA, with the minor adjustment that the pseudo-ID function uses a key shared with the L-node and H, rather than the BS.

Another major benefit of L-nodes making data transmissions only in response to the H-node sending a '*yes*' L-Control packet is that corrupt nodes will be unable to cause a resource depletion attack by sending false '*yes*' L-Control packets. The H-node will verify all incoming '*yes*' L-Control packets before sending one out in response. If an L-Control packet fails the verification, H can send a '*no*' L-Control packet to reset the cluster L-nodes to *idle*. The verification procedure is simply to see if the SourceID matches the expected pseudo-ID for any nodes in the cluster.

2.4.3.2. Inter-cluster Functionality

The relationship between H-nodes is very similar to the L-nodes in HSA. That is, they operate as independent peers, with no central authority. The main difference is that H-nodes can be assumed to be fitted with tamper-resistant hardware, thus source authentication for the H-Control packets is not supported.

H-nodes send H-Control packets to each other. Upon receipt of a *'yes'* L-control packet, an H-node will move to the *ready* state, and set the TimeToSend based on its propagation delay estimate T_P. When the current time matches TimeToSend, the H-node will send a *'yes'* L-control packet to its cluster. In response, the L-nodes will immediately send data packets.

PrevHop	$K_{Bj}\{\text{DataToSend} \mid \text{TimeToSend}\}$

Figure 9. H-Control Packet Format.

The H-Control packets consist of PrevHop, DataToSend and TimeToSend fields, which are the same as those used in HSA.

As in HSA, data packets are much longer than H-Control packets, containing the actual event data. Again, we will use 30 byte data packets for calculations.

Again, we will assume the delay between H-Control packets follows the exponential distribution with mean λ. An *idle* H-node will send *'no'* H-Control packets with a delay fitting the probability distribution, a *ready* H-node will send *'yes'* H-Control packets with the minimum delay that follows the distribution. A node moves to the ready state when one of two conditions have been met:

1) It has data to send.
2) It has received a *'yes'* control packet.

H-node clocks are synchronized. Each H-node has an estimate, T_P, for how long it takes a '*yes*' H-Control packet it sends to propagate through the network. An *idle* node with event data to send, will become a *ready* node, with Time To Send (TTS) set to the current time + T_P. An *idle* node that receives a '*yes*' control packet will use the TTS from the incoming '*yes*' packet.

As in HSA, we use an accelerated transmission delay of $\lambda/10$ for '*yes*' H-Control packets Each node will set $T_P = \frac{\lambda}{10} H$, where H is how many hops the H-node is from the far edge of the network. The same concerns for correct T_P estimation apply to TTSA as in HSA.

When the current time matches the TimeToSend, H-nodes broadcast '*yes*' L-Control packets to their clusters. The L-nodes send data packets to their cluster heads in response. All dummy traffic is filtered, and event traffic is aggregated. Each H-node creates a summary data packet which is then routed to the base station.

2.4.3.3. Estimating Overhead for TTSA

There are two main forms of overhead that we are concerned with minimizing: Transmission overhead and Latency. Transmission overhead is the amount of traffic that is required by the scheme that wouldn't be required otherwise. Latency is the sum of all delays imposed by the scheme over simple routing.

We will use the following notations to form the estimates:

- There are *n* L-nodes in the network.
- There are *N* H-nodes (clusters) in the network.

```
Algorithm for L-node functionality in Two-Tier Source Anonymity
Input: Mean delay between control packets λ
1.  φ := {}
2.  hasData := FALSE
3.  state := IDLE

IDLE state

→ 'no' L-Control Packet received from an L-node
1.  state := IDLE

→ 'yes' L-Control Packet received from an L-node
1.  ctrlDelay := MinDelay(φ, λ)
2.  set ControlTimer
3.  state := READY

→ 'no' L-Control Packet received from Cluster Head
1.  state := IDLE

→ 'yes' L-Control Packet received from Cluster Head
1.  send 'Dummy' Data Packet
2.  state := IDLE

→ ControlTimer fired
1.  φ := φ ∪ {ctrlDelay}
2.  send 'no' L-Control Packet
3.  ctrlDelay := RecoverMean(φ, λ)
4.  set ControlTimer
5.  state := IDLE

→ Sensor Event
1.  hasData := TRUE
2.  ctrlDelay := MinDelay(φ, λ)
3.  set ControlTimer
4.  state := READY

READY state

→ 'no' L-Control Packet received from an L-node
1.  state := READY

→ 'yes' L-Control Packet received from an L-node
1.  state := READY

→ 'no' L-Control Packet received from Cluster Head
1.  if hasData = TRUE then
2.      ctrlDelay := MinDelay(φ, λ)
3.      set ControlTimer
4.      state := READY
5.  else
6.      state := IDLE
7.  endif

→ 'yes' L-Control Packet received from Cluster Head
1.  if hasData = TRUE then
2.          send 'Event' Data Packet
3.  else
4.          send 'Dummy' Data Packet
5.  endif
6.  hasData := FALSE
7.  state := IDLE

→ L-ControlTimer fired
1.  φ := φ ∪ {ctrlDelay}
2.  send 'yes' L-Control Packet
3.  ctrlDelay := RecoverMean(φ, λ)
4.  set ControlTimer
5.  state := READY

→ Sensor Event
1.  set hasData := true
2.  state := READY
```

Figure 10. Algorithm describing behavior of L-nodes in scheme TTSA. Procedure MinDelay is given in Figure 2. φ maintains the history of L-Control Packet delays.

Algorithm for H-node functionality in Two-Tier Source Anonymity

Input: Distance H the node is from the far edge of the network (Hops)
Mean delay between control packets λ

```
1.  Tp := λ/10 H
2.  φ := {}
3.  hasData := FALSE
4.  state := IDLE
```

IDLE state

```
→ 'no' L-Control Packet received
→ 'no' H-Control Packet received
1.      state := IDLE

→ 'yes' L-Control Packet received
1.      if Verify(packet) = PASS then
2.          lCtrlDelay := Tp
3.          set L-ControlTimer
4.          hCtrlDelay := MinDelay(φ, λ)
5.          set H-ControlTimer
6.          state := READY
7.      else
8.          send 'no' L-Control Packet
9.          state := IDLE
10.     endif

→ 'yes' H-Control Packet received
1.      lCtrlDelay := packet[tts] - CurrentTime
2.      set L-ControlTimer
3.      hCtrlDelay := MinDelay(φ, λ)
4.      set H-ControlTimer
5.      state := READY

→ H-ControlTimer fired
1.      φ := φ ∪ {hCtrlDelay}
2.      send 'no' H-Control Packet
3.      hCtrlDelay := RecoverMean(φ, λ)
4.      set H-ControlTimer
5.      state := IDLE
```

READY state

```
→ 'no' L-Control Packet received
→ 'yes' L-Control Packet received
→ 'no' H-Control Packet received
→ 'yes' H-Control Packet received
→ 'Dummy' Data Packet received
1.      state := READY

→ 'Event' Data Packet received
1.      hasData := TRUE
2.      state := READY

→ H-ControlTimer fired
1.      φ := φ ∪ {hCtrlDelay}
2.      send 'yes' H-Control Packet
3.      hCtrlDelay := RecoverMean(φ, λ)
4.      set H-ControlTimer
5.      state := READY

→ L-ControlTimer fired
1.      send 'yes' L-Control Packet
2.      dataDelay := λ
3.      set DataTimer
4.      state := READY

→ DataTimer fired
1.      if hasData = TRUE then
2.          send 'Event' Data Packet
3.      else
4.          send 'Dummy' Data Packet
5.      endif
6.      state := IDLE
```

Figure 11. Algorithm describing behavior of nodes in scheme HSA. Procedure MinDelay is given in Figure 2. φ maintains the history of H-Control Packet delays.

- L-Nodes are an average of d hops from their cluster head.
- H-Nodes are an average of D hops from the BS.
- H-Nodes are an average of h hops from the far edge of the network.
- Every node sends a control packet every λ seconds, independent of whether it has data to send.
- Network lifetime in seconds is Φ.
- Data packets are δ bytes each.
- L-Control packets are β bytes
- H-Control packets are γ bytes
- The number of events during network lifetime is Ω.

For every event, a single data packet is generated, which must be routed to the nearest H-node and then to the BS, generating a total of $\Omega dD\delta$ bytes of traffic. Each L-node in the network sends an L-control packet every λ seconds, giving a total L-control traffic of $n\frac{\Phi}{\lambda}\beta$ bytes. Similarly, every H-node sends an H-control packet every λ seconds, giving a total H-control traffic of $N\frac{\Phi}{\lambda}\gamma$ bytes. For every event, all L-nodes generate a data packet which is routed to the base station, an average of d hops. L-node data traffic is thus $n\Omega\delta d$ bytes. H-nodes aggregate the results for their cluster and send a data packet to the BS. H-node data traffic is $N\Omega\delta D$ bytes.

Total traffic overhead generated is the sum of these terms less the payload, $n\frac{\Phi}{\lambda}\beta+N\frac{\Phi}{\lambda}\gamma+n\Omega\delta d+N\Omega\delta D-\Omega dD\delta$ bytes.

Latency is the time it takes for 'yes' L-control packets to propagate from the source L-node to its cluster head, 'yes' H-control packets to propagate among the H-nodes, and then for the data to be routed to the BS. It takes $\frac{\lambda}{10}d$ seconds to reach the first cluster head. Then, $\frac{\lambda}{10}h$ seconds to propagate through the cluster heads. In our implementation of the schemes in ns2, we allow λ seconds for the data to be routed from the L-nodes to the cluster head. Data is then immediately routed to the BS. The sum of these terms gives $\frac{\lambda}{10}d+\frac{\lambda}{10}h+\lambda$ seconds. This total ignores any delays imposed in the Two-Tier MAC layer (Section 2.3.4).

2.5. Analysis

2.5.1. Experimental Design Considerations

In order to make fair comparison between the schemes, the experiments were run using the same node deployments. In order to reduce the effect of randomness, multiple replicates are made using the same parameters, but different random number seed values. The data points used for evaluation are the average of the results from these replicates. As the number of replicates increases, the mean values will tend toward the true mean. While more replicates are desirable, the time involved in collecting the data must be weighed against the value of increasing accuracy. In our experiments, 10 replicates were made of all experiments.

2.5.2. Experimental Setup

In order to verify the efficiency of the schemes, they were implemented using the ns2 network simulator [21]. To ensure uniformity, nodes were placed using the concept of clusters, although no cluster heads were placed in FPR and HSA. The basic deployment strategy is 36 nodes per cluster in a grid pattern. Density was increased by randomly placing additional nodes within the area of the cluster, varying the number of nodes per cluster as {36, 47, 60, 72} without changing the physical area. The pseudorandom number facility of ns2 was utilized with predefined seed values to ensure repeatable "random" deployments.

Clusters were placed in a grid pattern, with h-nodes (TTSA only) in the center of the cluster. In all cases, the BS was located in the upper left corner of the placement area.

Ns2 is divided into two major components. The simulator itself is written in C++ and models the behavior of each node independently of the others. A script is written in TCL which determines node placement and schedules the events. Ns2 has a modular design. The functionality described above was implemented using the existing routing protocols and physical layer emulation.

2.5.3. Parameters

The network size was varied for all schemes over {1, 2, 4} clusters using the varying node densities described above. λ was varied to give similar latency ranges for the various schemes. The mean time between events σ was varied over the set {5, 10, 20, 30, 45, 60, 90, 120, 180, 1000}. When $\sigma=1000$, no events occur during the simulation. This gives a baseline result for the traffic generated by the schemes when idle. All simulations were run for 600 seconds, and 10 replicates were made with different random number seeds. The results of the 10 replicates were averaged.

The metrics measured were latency and traffic overhead. Latency is the time from the event occurrence until the message was received by the BS. Traffic overhead is the total number of bytes of traffic transmitted that aren't event data packets. That is, Control packets and Dummy packets are all traffic overhead.

2.5.4. Results

Unsurprisingly, in all cases, traffic overhead decreases as λ increases, and latency increases as λ increases. Since the various schemes perform differently for the same values of λ, the best way to compare them is to compare their cost (traffic overhead) vs. their performance (latency). The value of λ can be set based on these requirements.

Another important consideration is the expected mean time between events σ. As Figure 12 shows, FPR has a constant energy drain, regardless of whether there is data to protect. TTSA and HSA are reactive schemes, consuming much less energy when there is no data to protect. As σ increases, this difference become more significant. Compare Figure 13 and Figure 14. FPR is more efficient in a small network when events are frequent. This advantage is removed as the events decrease in frequency, and as the size of the network grows.

Figure 15 shows that for all schemes, traffic overhead increases as the size of the network grows. In all schemes, the trend is very linear, with FPR having the worst scalability, and TTSA having the best.

2.6. Conclusion

We have proposed efficient schemes to provide event source anonymity for a variety of sensor network configurations. Dummy traffic efficiency is greatly enhanced by making use of small control packets to synchronize the transmission of larger data packets.

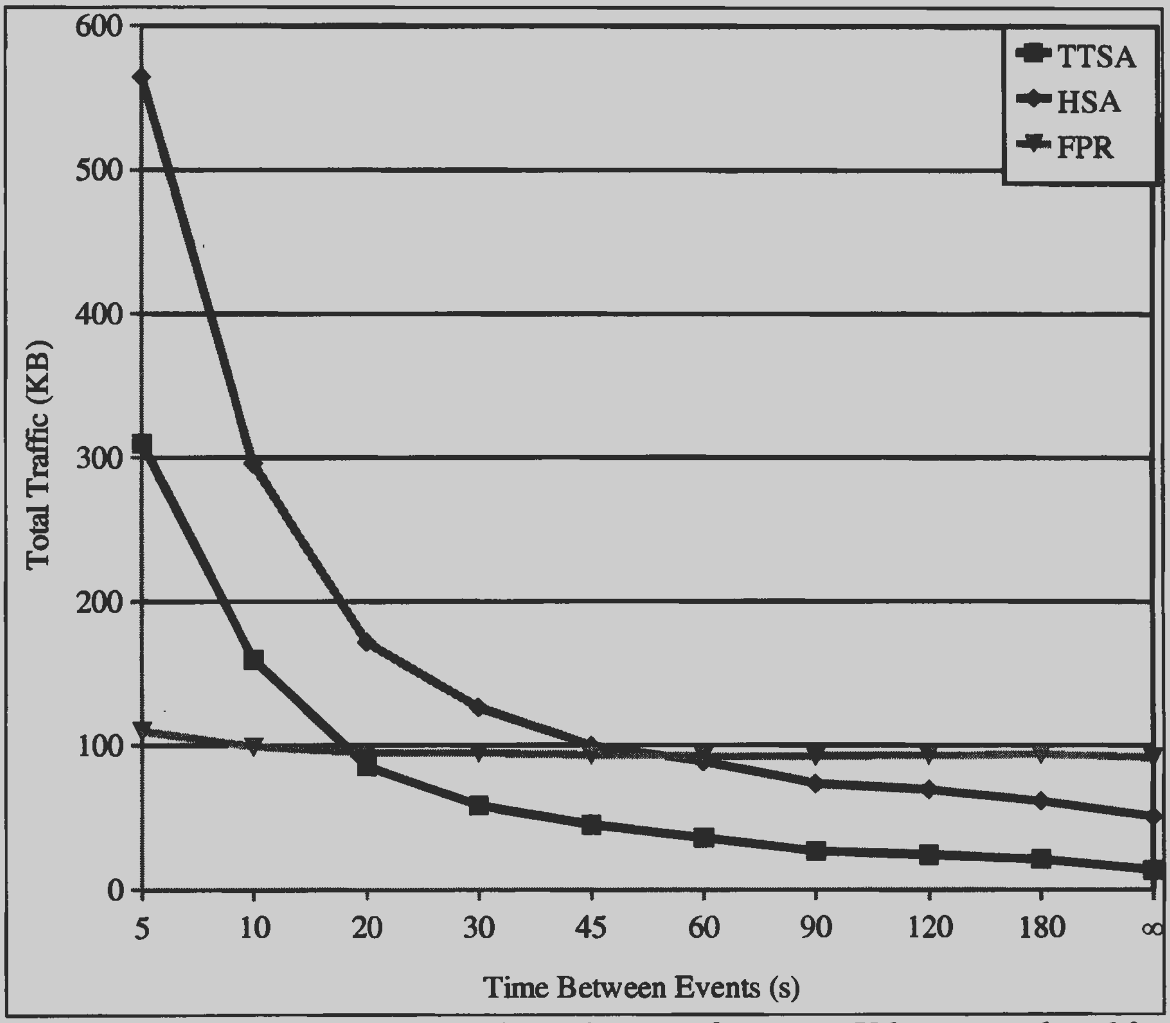

Figure 12. Total traffic generated over decreasing event frequency. Values were selected for λ yielding latency very close to 3s. Network deployment is 1 cluster of 36 L-nodes.

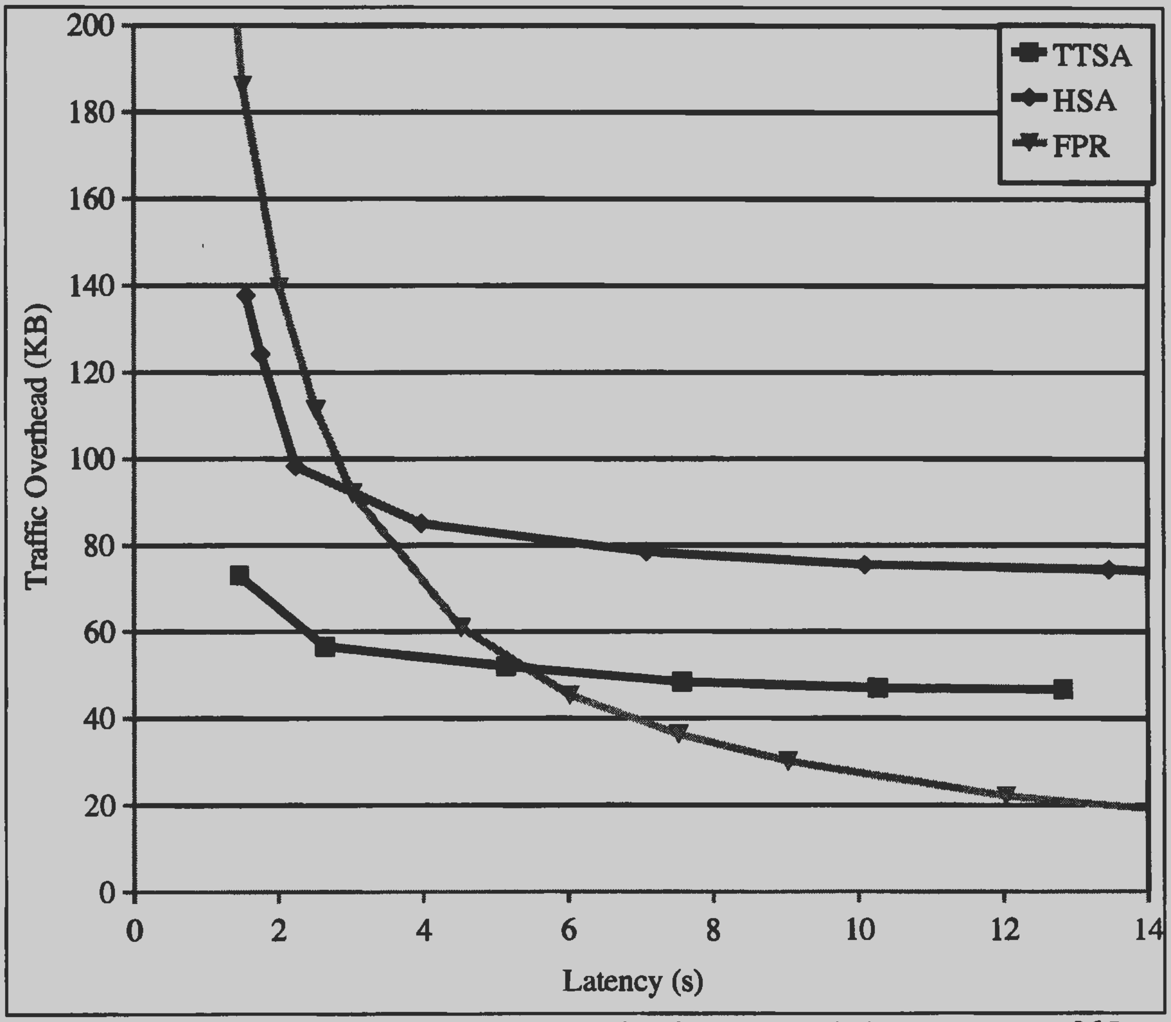

Figure 13. Relationship between Traffic overhead and Latency. 30s between events, 36 L-node network deployments.

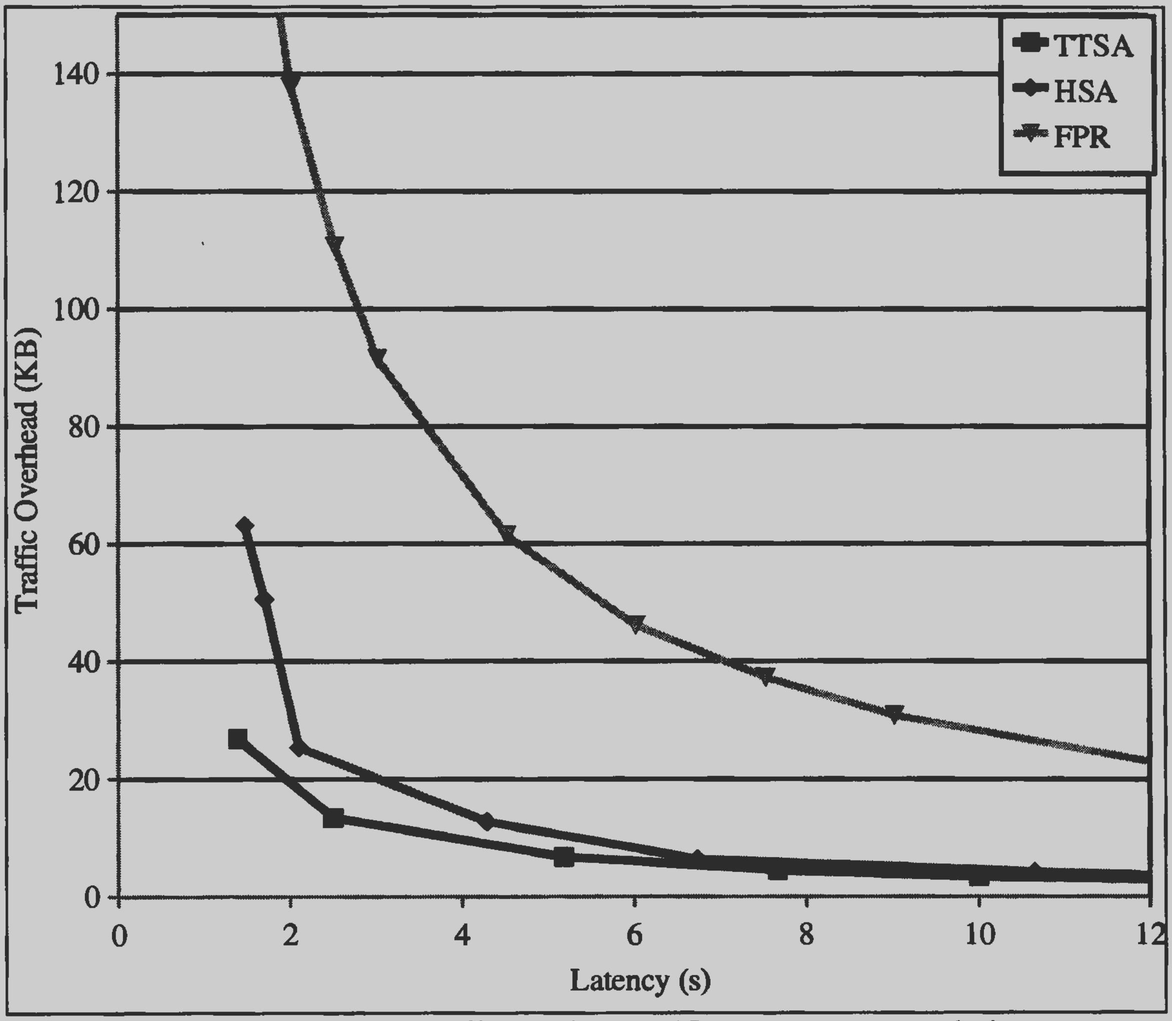

Figure 14. Relationship between Traffic overhead and Latency. No events during simulation, 36 L-node network deployment.

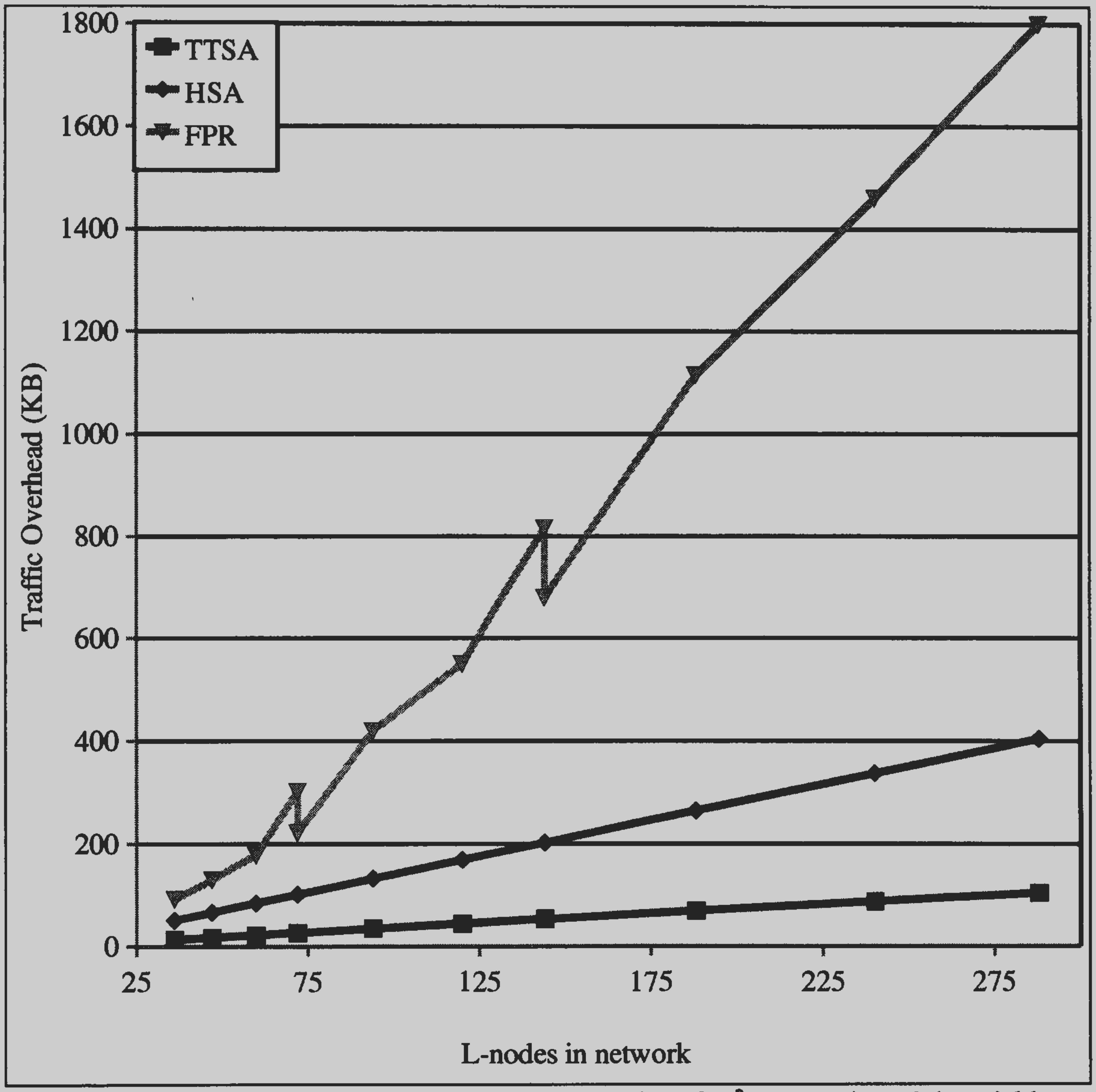

Figure 15. Traffic overhead as the network grows. Values for λ were selected that yield a latency very close to 3 seconds for all schemes. No events occurred during simulation.

The control packet based schemes are subject to a resource depletion attack by an insider sending false '*yes*' control packets. We show that in principle it is possible to mitigate this attack by detecting the source and revoking the malicious nodes. However, the Two-Tier scheme is by design robust against this attack and offers both greater security and scalability than the Homogeneous scheme.

A significant concern for efficient source anonymity is the latency imposed by the probabilistic delay used to thwart traffic analysis. By using small control packets to coordinate the transmission of dummy data, a much smaller probabilistic delay can be used with less traffic overhead. Our results show that reasonably low latency can be obtained with far less traffic overhead than the previous work while maintaining good scalability with large network deployments.

CHAPTER 3. DISTRIBUTED WIRELESS NODE IDENTIFICATION

3.1. Introduction

Security is an important and challenging issue in wireless sensor networks. A widely used attack model assumes that a sensor node does not have tamper resistant hardware (due to cost) and may be compromised in the field. A compromised node may be used to carry out various malicious attacks on the network. Several attacks on sensor nodes/networks have been studied, such as selective forwarding attack, wormhole attack, sinkhole attack, and Sybil attack [22].

In this chapter, we consider the malicious collision attack (see Section 1.2.2) that can be easily launched by a compromised (or hostile) sensor node. In a collision attack, an attacker node does not follow the medium access control protocol and cause collisions with neighbor node's transmissions by sending a short noise packet. This attack does not consume much energy of the attacker but can cause a lot of disruptions to the network operation. Due to the wireless broadcast nature, it is not trivial to identify the attacker.

In this chapter, we present a distributed scheme that is based on low-cost hardware and can effectively identify the source of a collision attack. Our scheme identifies the attacker by analyzing the physical-layer Received Signal Strength Index (RSSI) readings at neighbor nodes. RSSI readings are inherently unreliable due to the variability of the wireless medium. We overcome this unreliability through distributed sampling and centralized analysis of the RSSI readings. It has been shown that for multiple transmissions from a single source, the ratio of RSSI readings from neighbor nodes remains constant [23]. We leverage this fact to create unique fingerprints for nodes in a sensor network. The fingerprints are used for identifying the source of a collision attack with high confidence.

Most past work considered a homogeneous sensor network, where all nodes have the same (or similar) capabilities. In this work, we adopt a Heterogeneous Sensor Network (HSN) model that consists of a small number of powerful High-end sensors (H-nodes), in addition to a large number of small Low-end sensors (L-nodes). H-nodes have better capabilities than L-nodes in terms of communication, computation, energy supply, storage space, and other aspects. In our research, we take advantage of the strong capabilities of H-nodes for designing efficient and effective security schemes.

The rest of the chapter is organized as follows: We discuss the related work in Section 3.2., and describe the wireless fingerprinting framework in Section 3.3. In Section 3.4., we present several effective schemes for identifying the source of a collision attack, and we report the experimental results in Section 3.5. We discuss the results and conclude this chapter in Section 3.6.

3.2. Related Work

Demirbas and Song [23] developed a scheme for detecting the Sybil attack [22] by using the RSSI values from at least two detecting nodes. They showed that while the RSSI values for a given node vary greatly between transmissions, the ratio of RSSI values seen by two nodes for a given source is consistent. However, the goal in [23] is simply to determine whether two transmissions were from the same source, [23] did not present any practical techniques for determining the source of malicious transmission collisions in sensor networks. Furthermore, [23] only considered homogeneous sensor networks. Our work addresses a more difficult issue of identifying the source of a malicious collision.

Also, we considered a HSN and utilized more powerful H-nodes.

Law, *et al.*, [8] considered an attack where an outsider deploying a jamming network in the same area as the target network. They presented schemes for efficient jamming, with near 100% message suppression, while giving the jamming nodes a lifetime similar to the target network. Suggestions for more robust MAC layer protocols are given in [8].

A number of literatures have discussed methods for wireless fingerprinting by analyzing characteristics of the radio signal. Some are discussed below:

- Frequency shift – Due to the cost of manufacturing, every radio transmits at a slightly different frequency [24], and this can be used to identify a radio device.
- Transients – During power up and power down, wireless radios emit a noise signal. The noise signals are referred to as transients and are unique to each physical device [25].
- Signal strength – A closer node usually has a stronger signal than one far away [23] when similar transmission powers are used.
- Clock skew – Due to manufacturing reasons, each node has a unique clock skew, and the skew can be used to identify a node [26].

Techniques (e.g., those in [24], [25], [26]) relying on analysis of the physical radio signal typically require expensive hardware to obtain the necessary accuracy. However, the RSSI is a notable exception and the RSSI value is available in many wireless devices. On the other hand, RSSI is also unreliable for two reasons: 1) A common energy saving technique is to vary the transmission power to only the level needed for reaching the desired neighbor. If sensors dynamically change their transmission powers, the RSSI value itself is not very useful for node identification. 2) The signal strength of a transmission also varies due to environmental conditions, and can be unreliable even if the transmission power is fixed.

Faria and Cheriton [27] developed a RSSI based fingerprinting scheme, in which a fingerprint is the RSSI values recorded by multiple Access Points. Similarly to [23], they want to decide whether multiple transmissions came from the same source. In order to combat the effects of varying transmission power, the difference between RSSI readings from the same transmissions is used to determine an attacker. However, the actual differences between RSSI readings vary a lot and are not reliable. In our scheme, we use the fact that the ratio of RSSIs from two observers remains constant, regardless of the source transmission power. Yedavalli, *et al.* [28] utilized RSSI for localization. The scheme in [28] is referred to as *Ecolocation*, and it is based on the distance-based rank-ordering by detector nodes with known locations. The assumption is that RSSI is correlated with distance, and the rank-ordering is determined by the location of the unknown node.

3.3. Wireless Fingerprinting Framework

3.3.1. Network Model

After sensor deployment, clusters are formed in a HSN. An efficient cluster formation scheme for HSNs can be found in [29]. Each cluster contains one H-node and a number of L-nodes, and the H-node is the cluster head. L-nodes respond to queries from and send data to its cluster head. A cluster head (H-node) aggregates data and then send it to the base station. An H-node is a more powerful node, and can communicate directly to all (or most) L-nodes in its cluster. L-nodes are small, low-power nodes which send data to the cluster head via multi-hop communications.

3.3.2. Attack Model

An L-node may be captured and compromised, and then all the data, software and security materials will be revealed. Zhu, *et al.* [3] suggested that there is a minimum time for an adversary to compromise a sensor node, and within the time period the network is assumed to be secure. In this paper, we make the same assumption as [3]. In addition, we assume that H-nodes are trustworthy. For example, H-nodes may be installed with tamper-resistant hardware. This is a reasonable assumption for powerful H-nodes.

Our main goal is to identify the source of a malicious collision attack, where an adversarial node makes transmissions timed to cause collisions with legitimate neighbor communications, for the purpose of disrupting traffic. For example, suppose the IEEE 802.11 MAC is used, and node u wants to send a packet to a neighbor node v. Based on the RTS/CTS exchanges, a neighboring adversarial node x knows the timing of u to transmit the data packet, and x can transmit a noise that overlaps with the data packet and hence cause collisions. The malicious collision attack also allows an attacker to carry out the selective forwarding attack [22] for routes that it isn't actually on. Since the attacker may not follow any protocol, we make no assumptions about the format of the collision packet other than that it has measurable signal strength to all neighbors.

3.3.3. Building the Fingerprints

In this paper, we propose a scheme that can identify the source of a malicious collision attack by using the RSSI readings. The scheme is based on the fact that the ratios of the RSSI readings between two neighbor nodes remain the same (or very close) for different transmissions from the same source, even the transmissions use different powers. Denote $R_u(1)$ as the RSSI reading at node u for transmission #1. Suppose that neighbor nodes u and v record the RSSIs from two transmissions, if the following result holds,

$$\frac{R_u(1)}{R_v(1)}=\frac{R_u(2)}{R_v(2)} \tag{3}$$

then we can claim that the same source node transmitted packet #1 and #2.

The RSSI ratio in Equation (3) is a fingerprint of a node. In this subsection, we discuss how to build the fingerprints for node identification. During the initiation phase (assumed no attacks), all L-nodes send *hello* messages with a sequence number by using the same power. Each L-node records the RSSIs of neighbors' *hello* messages and the corresponding sequence numbers. Then the RSSI values are sent to the cluster head (denoted as H).

A sample set of RSSI readings [30] is given in Table 1. These RSSI readings are data from actual 802.11 wireless transmissions collected on the Orbit test-bed [31] at Dartmouth College. The experiment layout is shown in Figure 16, where 29 nodes are deployed in a grid of 8x8 cells, and the cell length is 1 meter. Xs denote nodes, and Os denote noise generators. The node is labeled by its coordinates in the grid, e.g., 1-2 is the node locates at row 1 and column 2. In Table 1, the first row is the node label *x-y*. Rows 2 – 6 list the RSSI readings of five *hello* messages sent by the node at location 1-2, as recorded by all of its neighbors.

The set of RSSIs for a given transmission from multiple neighbors are referred to as a

Table 1. Partial sample RSSI readings for five hello messages sent by node 1-2.

	1-4	1-6	1-8	2-1	2-5	3-2	3-4	3-6	3-8	4-1	4-3	4-5	4-7	5-2
Report0	38	19	21	23	27	35	25	20	22	26	27	29	11	31
Report1	38	18	20	21	26	34	24	18	22	26	27	27	11	30
Report2	37	14	18	19	24	30	22	16	19	23	24	22	7	27
Report3	37	12	16	16	20	32	20	14	16	21	21	21	6	25
Report4	36	11	15	16	21	32	20	13	16	20	22	22	6	25
Mean	37.2	14.8	18	19	23.6	32.6	22.2	16.2	19	23.2	24.2	24.2	8.2	27.6

report. For example, in Table 1, Report1 includes all RSSIs in row 2. When a node logs a RSSI, the timestamp is recorded as well. The timestamp is sent to H along with the RSSI value. The fingerprint of a node is the set of all reports corresponding to *hello* messages sent by that node.

To reduce the overhead of RSSI fingerprint generation, the following schemes may be used:

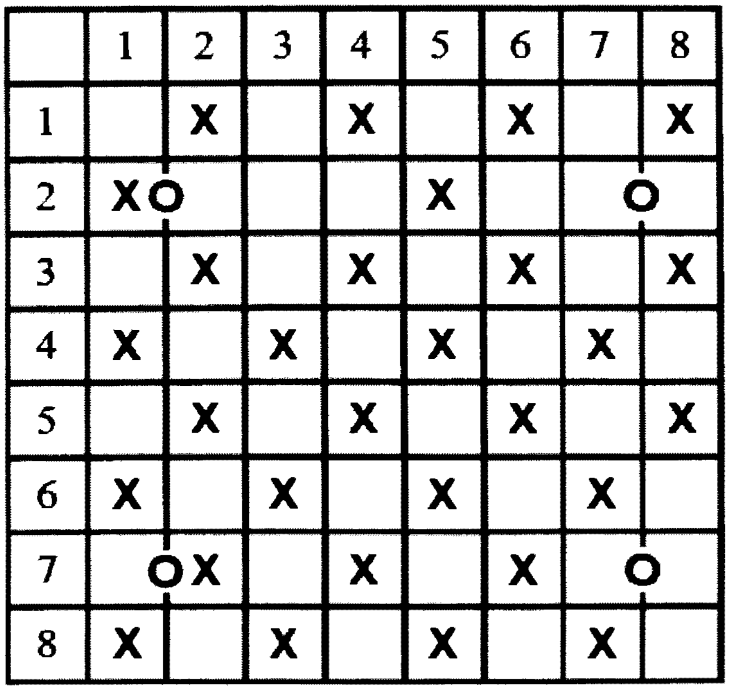

Figure 16. The network topology. 'X' denotes a node, 'O' denotes a noise transmitter.

1) If the IEEE 802.11 MAC is used, then the RTS/CTS packets can be used for recording the RSSIs and hence generating the fingerprint of a node.

2) If there are no MAC control packets being sent before the data packet (e.g., TDMA is used), then the RSSI can be obtained from the packet header. The header should be received by all neighbors so that a neighbor node knows whether it is the intended recipient.

In both cases, no dedicated packets (e.g., *hello* messages) are used to generate the RSSI fingerprints, and hence the communication overhead is reduced.

After receiving all the reports, the cluster head H will compare the reports based on the timestamp to make sure RSSIs from the transmission is used to build the fingerprint. When a collision attack happens, there are two transmissions (the legitimate packet and the collision packet) occurring simultaneously. Suppose node *u* transmits to *v*, and node *x* causes a collision. When node *v* detects a collision, it assumes that there is a collision attack. However, *v* does not know who the attacker is. After detecting a collision attack, node *v* sends to all of its 2-hop neighbors an *alarm* message, which includes the legitimate sender ID *u* and the time *t* of the collision attack. Each 1-hop neighbor of node *u* should hear the legitimate transmission from *u* (or the collision). Hence, *u*'s 1-hop neighbors will not respond to the *alarm* message (i.e., do not report to H). When other nodes receive the *alarm* message, each sends to the cluster head H a *report* message that includes the RSSI and the timestamp of a transmission around the time *t*. H will use the timestamps to correlate the readings. After collecting the RSSI readings from the *report* messages, H will build a RSSI ratio, and compare it with the RSSI ratio fingerprint. The node that has the closest match is considered as the attacker. We discuss the details of several identification schemes in Section 3.4.

3.4. Effective Schemes for Identifying the Attacker

3.4.1. The Average RSSI Value Scheme

To reduce the communication overhead of building fingerprints, each L-node should aggregate the RSSI readings from all its neighbors and only send a single report to H. A simple way to do this is for each L-node to take the average RSSI of multiple *hello* messages and send the average instead of the individual RSSIs. To minimize the variations of RSSI readings, *hello* messages are transmitted with a constant power. The above scheme is referred to as the Average RSSI Value (ARV) scheme. For example, for the RSSI data in Table 1, each L-node computes the average of the RSSI values, as listed in the last row, and sends the averages to H.

When a collision attack happens, H collects the *event reports* from L-nodes, and builds the RSSI ratio of the attacker. Then H compares the attacker's RSSI ratio with that of each candidate node *y* (neighbors of node *v*). A score is used to indicate the magnitude of the difference between the RSSI ratios. The RSSI ratios are computed for every pair of nodes *i* and *j* that have valid RSSI readings stored in the fingerprint and are listed the *event reports*. A candidate *y*'s score is the average of the differences of RSSI ratios for all nodes *i* and *j*. The candidate *y* with the lowest score is identified as the source of the collision attack. Figure 17 lists the ARV scheme. In Figure 17, *Report* is the *event report* being analyzed, FP*y* is the fingerprint of node *y*, and *i*, *j*, *y* are L-nodes.

Algorithm to compute the ARV scheme

```
1.  foreach candidate node y
2.      $score[y] := \infty$
3.      foreach $i \varepsilon$(FPy and Report)
4.          $scorej := 0$
5.          $n := 0$
6.          foreach $j \varepsilon$(FPy and Report), $j \neq i$
7.              $scorej \mathrel{+}= \left| \frac{Report[j]}{Report[i]} - \frac{FPy[j]}{FPy[i]} \right|$
8.              $n := n + 1$
9.          endfor
10.         $scorej := scorej / n$
11.         if $scorej < score[y]$ then
12.             $score[y] = scorej$
13.         endif
14.     endfor
15. endfor
```

Figure 17. Algorithm to find the transmission source using the Average RSSI Value scheme.

3.4.2. The Constraint-based Average RSSI Value Scheme

Ecolocation [8] uses the concept of constraints to estimate a node's position. In [8], a constraint is given by the distance-based rank ordering of a pair of neighbors. I.e., if neighbor *i* is closer than neighbor *j* to a node *u*, then the **constraint match** requires that the distance between *u* and *i* is less than that between *u* and *j*. In [8], a set of constraints is used to estimate the location of a node.

We apply the constraint technique to the ARV scheme, and refer to this new scheme as Constraint-based Average RSSI Value (CARV) scheme. During network initiation, each L-node collects RSSI readings from their neighbors, and sends the average RSSI to its cluster head H. When a collision attack is detected, neighbor L-nodes send RSSI reports to H for analysis.

We define a constraint function c(*s*, *t*) as follows:

$$c(s,t)=\begin{bmatrix} 1 \; if\,(s<t) \\ 0 \; if\,(s=t) \\ -1 \; if\,(s>t) \end{bmatrix} \tag{4}$$

The constraint function c(s, t) compares two input values s and t and determines which is larger. A constraint is matched if and only if for two pairs of RSSI values (s_1, t_1) and (s_2, t_2), the following holds:

$$c(s_1,t_1)=c(s_2,t_2) \tag{5}$$

Constraints are calculated once for every pair of nodes i and j that have RSSI readings for each candidate node y in both the fingerprint and the *event report*. For each candidate node y, H computes a **score** that is the number of matched constraints minus the number of violated constraints. The candidate node y with the highest score is selected to be the best match. See Figure 18.

Algorithm to compute the CARV scheme

```
1.  foreach candidate node $y$
2.      $score[y] := 0$
3.      foreach $i \in$(FPy and Report)
4.          foreach $j \in$(FPy and Report), $j>i$
5.              if c(FPy[$i$], Fpy[$j$]) = c(Report[$i$], Report[$j$]) then
6.                  $score[y] := score[y] + 1$
7.              else
8.                  $score[y] := score[y] - 1$
9.              endif
10.         endfor
11.     endfor
12. endfor
```

Figure 18. Algorithm to find the transmission source using the constraint-based scheme.

3.4.3. The Hybrid Scheme

The ARV scheme is based on the fact that the RSSI ratio between two detector nodes for a given node should remain the same. The CARV scheme relaxes this requirement, and only considers if the one RSSI is larger than the other, instead of considering the actual ratio value. In this subsection, we present a Hybrid Average RSSI Value (HARV) scheme. The HARV scheme is similar to CARV scheme, but we change the way constraints are verified. As shown in Figure 19, only line 5 is different from CARV (Figure 18). A constraint is matched if the difference of two RSSI ratios is less than a threshold ε. In [23], a threshold of 5σ was used in the Sybil attack detection experiments, where σ is the Standard Deviation of the difference in RSSI ratios of consecutive messages as recorded by two detector nodes. In their experiments, $5\sigma = 0.5$. We conducted experiments by varying ε, and found 0.5 to be a good value. The results shown in Section 3.5. were collected with the parameter $\varepsilon=0.5$. Again, the candidate node y with the highest score is selected as the source of the attack.

```
Algorithm to compute the hybrid scheme
1.   foreach candidate node $y$
2.       $score[y] := 0$
3.       foreach $i \in$(FPy and Report)
4.           foreach $j \in$(FPy and Report), $j>i$
5.               if $\left|\frac{Report[i]}{Report[j]} - \frac{FPy[i]}{FPy[j]}\right| < \varepsilon$ then
6.                   $score[y] := score[y] + 1$
7.               else
8.                   $score[y] := score[y] - 1$
9.               endif
10.          endfor
11.      endfor
12.  endfor
```

Figure 19. Algorithm to find transmission source using the threshold-constraint hybrid scheme.

3.4.4. The Localization-based Scheme

In this subsection, we present a Localization-based (LOC) scheme. The LOC scheme is an implementation of the location estimation scheme described in [32]. If each L-node knows its own location, then it is possible to estimate the location of the source of any transmission that is observed by at least four L-nodes. We can estimate the location by solving the following set of equations for *x* and *y*:

$$\begin{aligned}(x-x_i)^2+(y-y_i)^2 &= \left(\frac{R_i}{R_j}\right)^{\frac{1}{\alpha}}\left((x-x_j)^2+(y-y_j)^2\right)\\ &= \left(\frac{R_i}{R_k}\right)^{\frac{1}{\alpha}}\left((x-x_k)^2+(y-y_k)^2\right)\\ &= \left(\frac{R_i}{R_l}\right)^{\frac{1}{\alpha}}\left((x-x_l)^2+(y-y_l)^2\right)\end{aligned} \tag{6}$$

where R_i is the RSSI recorded by node *i; i, j, k,* and *l* are the L-nodes that observed the transmission made by the node located at (x, y); and α is the distance-power gradient.

H obtains the R_i from *event reports*, and then derives the location of the source node based on Equation (6). The L-node closest to the source location is considered to be the attacker. One distinct advantage of this scheme is that it does not rely on any fingerprint. The location is derived based on a single, independent transmission. Also, it is not the identity of the node that is revealed, but the location.

3.5. Performance Evaluation

In this Section, we present the performance evaluation of the four schemes given in Section 3.4. We utilize the RSSI data collected by Kaul, *et al.* [30] on the ORBIT test bed [31]. The test-bed topology is shown in Figure 16, where 29 nodes were deployed in a grid of 8x8 cells with 1 meter cell size. Since the data was collected by other researcher for independent work, we had no control over the network deployment. While a simulated network environment would offer greater control over the deployment strategy, live trace data gives a much more realistic picture of the effectiveness of our schemes.

Five sets of RSSI data were collected, varying the power of the noise from -20 dbm to 0 dbm in an increment of 5 dbm. Each node made 300 transmissions; the RSSI for each transmission was recorded by the remaining nodes. We use the first 100 transmissions as our training set, and the remaining 200 transmissions as our data set. In our evaluations, 20 transmissions per node were used for testing. The 20 transmissions were chosen by taking every 10th message from transmission 100 to 300. A close inspection of the generated fingerprints shows that the data set includes one node that was failed. That is, none of the messages it was supposed to send were captured by any of the other nodes. Further, there are three nodes that have poor quality fingerprints. These four nodes cause the vast majority of the inaccuracy in the test. Rather than removing unfavorable data, we present results based on the full data set.

3.5.1. Finding the Optimal Size of Training Set

A parser was written in C++ to build the fingerprints and analyze the data against the test set. In order to compare the various schemes discussed in Section 3.4., trials were run by varying the size of the training set. A training set of size *n* used the first *n* messages for training. The test set remained the same for an accurate comparison. Figure 20 plots the accuracy of identifying the attacker vs. the size of the training set, and it shows that the training set size does not have significant impact on the accuracy. Note the accuracy varies between 0.86 and 0.90.

3.5.2. A Closer Look at RSSI Ratios

It is interesting to note that the accuracy doesn't increase with the size of the training set. In order to understand why, another experiment was run to compare the RSSI ratios

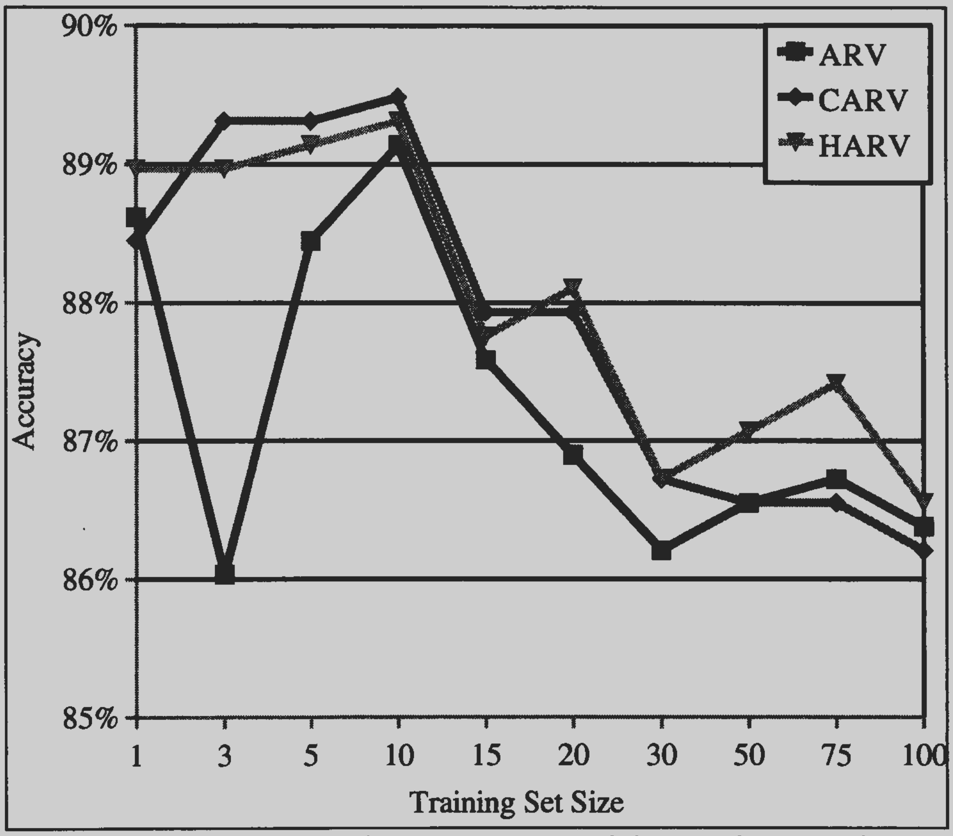

Figure 20. Source identification accuracy vs. training set size. Results collected from data set dbm -20.

from one transmission to the next. This was, in effect, a verification of the claims made by Demirbas and Song [23] that the difference in RSSI ratios followed the gaussian PDF, with a very small standard deviation.

As an example, source i sends two packets, 1 and 2, the RSSI of which are recorded by detector nodes u and v. The difference in RSSI ratios for packets 1 and 2 is computed with (7) where i=1.

$$\textit{difference}_i = \left| \frac{R_u(i)}{R_v(i)} - \frac{R_u(i+1)}{R_v(i+1)} \right| \tag{7}$$

The means in Figure 21 and Table 1 are the means of the differences calculated using (7). Once the mean is found, the standard deviation is computed in the usual way:

$$stddev=\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(difference_i-mean\right)^2} \quad (8)$$

As Figure 21 shows, the mean was consistently close to 0, but the standard deviation varied greatly depending on the amount of ambient noise. The source node 1-2 was selected because it had complete readings for all 5 data sets. As the noise increases, the number of missing packets increases. This leads to incomplete readings in the noisier data sets, with individual sources having a standard deviation of RSSI ratios as high as 8. The conclusion is that RSSI ratios might be less consistent than previously thought. The consistency of RSSI ratios degrades considerably as the amount of ambient noise increases.

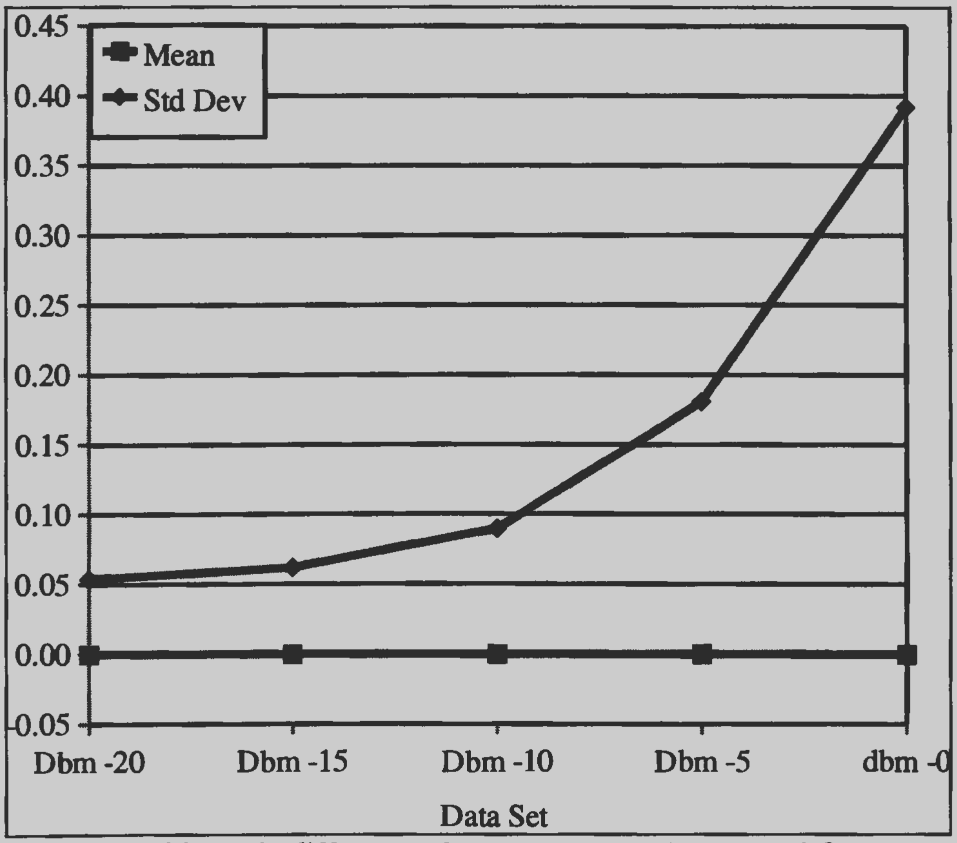

Figure 21. RSSI Ratio differences for messages sent by source 1-2.

Table 2 shows a sample of results from the dbm-20 data set, which is the least noisy of the sets available. Even in this noise-free set, the standard deviation varies quite a bit over the various source nodes.

Table 2. Sample data for RSSI Ratio analysis – Collected using data set dbm -20.

Source	Detector 1	Detector 2	Mean	Standard Deviation
1-2	4-3	8-5	-0.0001	0.0538
2-5	4-3	8-5	-0.0010	0.2765
4-1	4-3	8-5	-0.0004	0.1129
5-4	5-6	3-4	-0.0002	0.0461
6-5	3-2	4-5	0.0001	0.0664

3.5.3. Evaluation of the LOC Scheme

Since the LOC scheme does not rely on RSSI fingerprints, it is considered separately from the other three schemes. The algorithm used to solve Equation (6) is given in Figure 22, and is explained below: First, we rearrange the equations and move the terms to one side. We seek to find values for x and y that minimize the error. Four detector nodes i, j, k, l are chosen at random, and α is set to 2. The locations of the four detector nodes are given as (x_i, y_i), (x_j, y_j), (x_k, y_k), (x_l, y_l), respectively. For each transmission, the algorithm is run to estimate the sender's location, and the L-node closest to the computed location (in terms of Euclidean distance) is considered as the sender.

As shown in Figure 23, the performance of the LOC scheme is quite poor. In order to better understand why the performance was poor, we examine the average error of the resulting coordinates versus the actual coordinates of the source for each transmission. The results are listed in Error: Reference source not found. For all data sets, the average

Algorithm to compute LOC

```
1.  $minErr := \infty$
2.  for $x = 0$ to 10 step 0.1
3.      for $y = 0$ to 10 step 0.1
            $err := \left| (x-x_i)^2+(y-y_i)^2-\left(\frac{R_i}{R_j}\right)^{\left(\frac{1}{\alpha}\right)}\left((x-x_j)^2+(y-y_j)^2\right) \right|$
4.              $+ \left| (x-x_i)^2+(y-y_i)^2-\left(\frac{R_i}{R_k}\right)^{\left(\frac{1}{\alpha}\right)}\left((x-x_k)^2+(y-y_k)^2\right) \right|$
                $+ \left| (x-x_i)^2+(y-y_i)^2-\left(\frac{R_i}{R_l}\right)^{\left(\frac{1}{\alpha}\right)}\left((x-x_l)^2+(y-y_l)^2\right) \right|$
5.          if $err < minErr$ then
6.              $bestX := x$
7.              $bestY := y$
8.              $minErr := err$
9.          endif
10.     endfor
11. endfor
```

Figure 22. Algorithm to find the physical location of the source of a transmission.

localization error was greater than 3 meters. This is consistent with the results in [33], which found that a median error of 10 feet (3 meters) can be expected with localization based on IEEE 802.11 devices. Given that the nodes are placed on a 1-meter grid, an average error of over 3 meters renders the LOC scheme useless.

Table 3. Localization errors.

Dataset	Error (meters)	Standard Deviation
dbm -20	3.88	1.89
dbm -15	3.75	1.82
dbm -10	3.85	1.88
dbm -5	3.48	1.87
dbm -0	3.18	1.87

3.5.4. The Accuracy of the Schemes

We evaluate the accuracy of the four schemes under different noise levels. The accuracy is defined as the percentage of a scheme correctly identifying the source node of a transmission. Specifically, we tested the accuracy of the four schemes under five different ambient (background) noise levels, from -20 dbm to 0 dbm, with an increase of 5dbm. The results are reported in Figure 23. As we can see the HARV performs better than other schemes, especially when the noise level increases. HARV has accuracy between 0.8 and 0.9. ARV and CARV perform reasonably well, but degrade as ambient noise increases. LOC performed poorly in all cases.

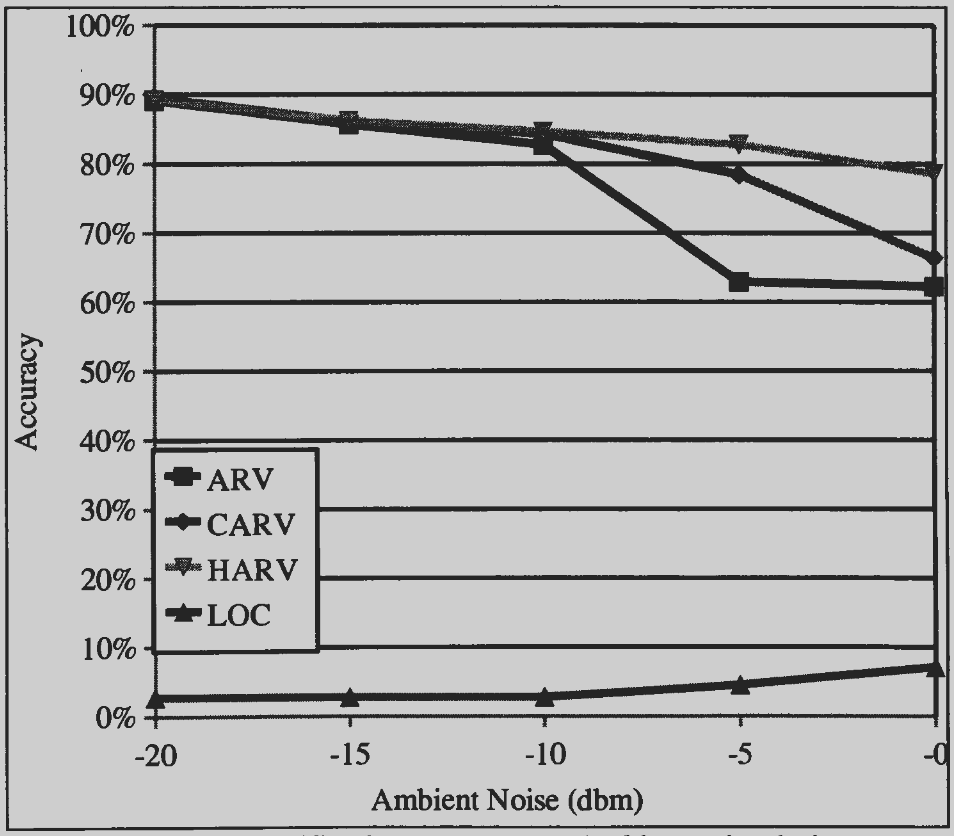

Figure 23. Source identification accuracy vs. Ambient noise during experiment. Data collected using training set size=10.

3.5.5. Performance During Increasing Ambient Noise

During a long-term deployment scenario, it is reasonable to expect that the ambient noise will vary over time. This is particularly true if another (possibly hostile) wireless network is deployed in the area. In order to test the effectiveness of the schemes through changing noise conditions, fingerprints were constructed using the dbm -20 data set. Node identification was attempted using these fingerprints for all data sets with increasing ambient noise. LOC was not considered since it does not rely on fingerprints. Again, a training set of 10 messages was used. The results are given in Figure 24.

As expected, in all cases the performance degrades as ambient noise increases. While ARV degrades drastically, CARV degrades gracefully, maintaining accuracy greater than 80% in all but the noisiest data set. The performance of HARV is between the two. The results suggest that for long-term deployment scenarios, CARV is the best candidate.

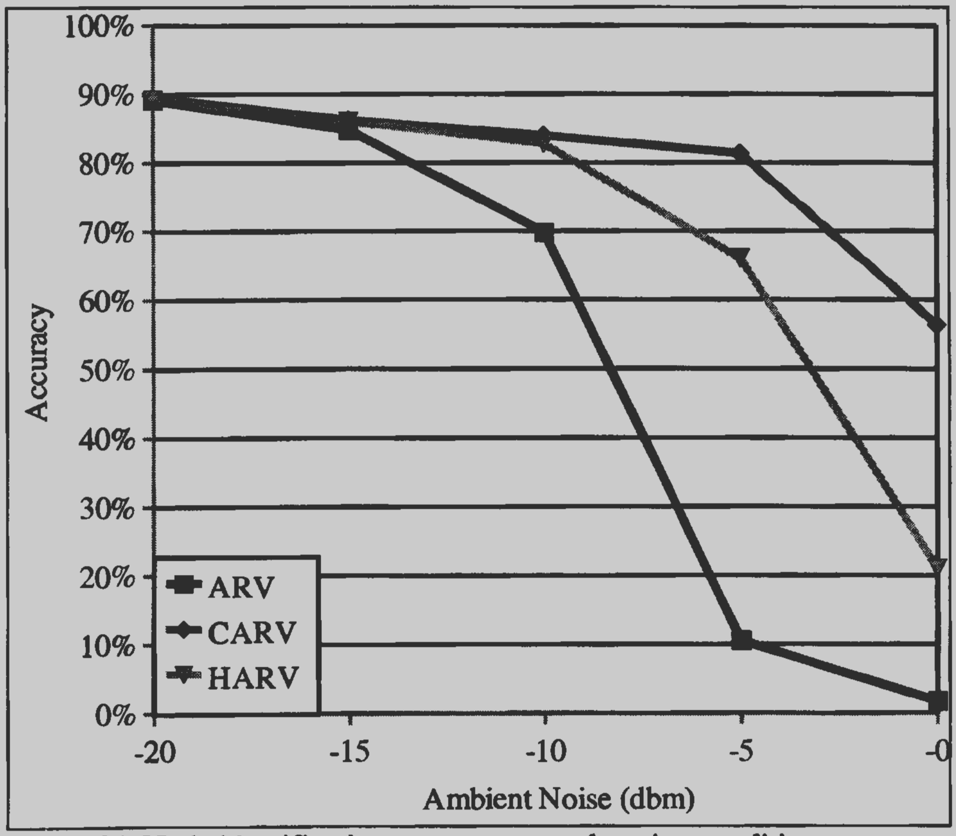

Figure 24. Node identification accuracy over changing conditions. Fingerprints were constructed from dbm -20 data set. A training set of 10 messages was used.

3.6. Conclusion

In this chapter, we studied the malicious collision attack in wireless sensor networks. The attack can be easily launched by a compromised or hostile node by timing its transmission of a short noise and cause a collision with neighbor's transmission. This attack does not consume much energy of the attacker but can seriously disrupt communications in the network. Due to the wireless broadcast nature, it is not trivial to identify the attacker. In this paper, we proposed three effective schemes (ARV, CARV, and HARV) for identifying the source of the collision attack. The schemes only require low-cost hardware and very suitable for small sensor nodes. The schemes are based on the physical-layer Received Signal Strength Index (RSSI) readings and utilized the fact that the ratio of RSSIs from two neighbors is consistent for the same send. One of the schemes – the CARV scheme degrades gracefully as the ambient noise increases over time. We evaluated the performance of the schemes based on RSSI data collected from real wireless transmissions. Our results showed that the three schemes can correctly identify the source of a collision attack with greater than 85% accuracy. Our results also showed that the traditional localization scheme performed poorly.

3.7. Acknowledgement

This research was supported in part by the US National Science Foundation (NSF) under grants CNS-0721907 and CNS-0709268, and the Army Research Office under grants W911NF-07-1-0250 and W911NF-08-1-0334.

CHAPTER 4. CONCLUSION

The two main contributions of this work are an efficient scheme for providing event source anonymity, and an effective scheme for finding the source of spurious transmissions.

A significant concern for efficient source anonymity is the latency imposed by the probabilistic delay used to thwart traffic analysis. By using small control packets to coordinate the transmission of dummy data, a much smaller probabilistic delay can be used with less traffic overhead. Our results show that reasonably low latency can be obtained with far less traffic overhead than the previous work while maintaining good scalability with large network deployments.

The control packet based schemes are subject to a resource depletion attack by an insider sending false '*yes*' control packets. We show that in principle it is possible to mitigate this attack by detecting the source and revoking the malicious nodes.

The source detection schemes are based on the physical-layer Received Signal Strength Index (RSSI) readings and utilize the fact that the ratio of RSSIs from two neighbors is consistent for the same sender.

We evaluated the schemes under a variety of ambient noise conditions. One of the schemes – the CARV scheme, degrades gracefully as the ambient noise increases over time. We evaluated the performance of the schemes based on RSSI data collected from real wireless transmissions. Our results showed that the three schemes can correctly identify the source of a collision attack with greater than 85% accuracy.

REFERENCES

[1] X. Du, M. Guizani, Y. Xiao, and H. Chen, "Two Tier Secure Routing Protocol for Heterogeneous Sensor Networks," in *IEEE Transactions on Wireless Communications,* pp. 3395-3401, 2007.

[2] Y. Yang, X. Wang, S. Zhu, and G. Cao, "Distributed Software-based Attestation for Node Compromise Detection in Sensor Networks," in *Reliable Distributed Systems, 2007,* pp. 219-230, 2007.

[3] S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks," in *Proceedings of the 10th ACM conference on Computer and communications security,* pp. 62-72, 2003.

[4] M. Shao, Y. Yang, S. Zhu, and G. Cao, "Towards Statistically Strong Source Anonymity for Sensor Networks," in *IEEE INFOCOM 2008,* pp. 51-55, 2008.

[5] Y. Yang, M. Shao, S. Zhu, B. Urgaonkar, and G. Cao, "Towards Event Source Unobservability with Minimum Network Traffic in Sensor Networks," in *Conference on Wireless Network Security,* pp. 77-88, 2008.

[6] "Crossbow Technology," http://www.xbow.com, accessed September 9, 2009.

[7] C. Castlluccia, E. Mykletun, and G. Tsudik, "Efficient Aggregation of encrypted data in Wireless Sensor Networks," in *MobiQuitous 2005,* pp. 109-117, 2005.

[8] Y. Law, L. Hoesel, J. Doumen, P. Hartel, and P. Havinga, "Energy-Efficient Link-Layer Jamming Attacks Against Wireless Sensor Network MAC Protocols," in *Proceedings of the 3rd ACM Workshop on Security of Ad hoc and Sensor Networks,* pp. 76-88, 2005.

[9] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler, "SPINS: Security Protocols for Sensor Networks," in *Wireless Networks*, vol. 8, Issue 5, pp. 521-534, 2002.

[10] Y. Ouyang, Z. Le, Y. Xu, N. Triandopoulos, S. Zhang, J. Ford, and F. Makedon, "Providing Anonymity in Wireless Sensor Networks," in *IEEE Conference on Pervasive Services,* pp. 145-148, 2007.

[11] K. Mehta, D. Lio, and M. Wright, "Location Privacy in Sensor Networks Against a Global Eavesdropper," in *IEEE International Conference on Network Protocols,* pp. 314-323, 2007.

[12] V. Mhatre, and C. Rosenberg, "Homogeneous vs Heterogeneous Clustered Sensor Networks: A Comparative Study," in *2004 IEEE International Conference on Communications,* pp. 3646-3651, 2004.

[13] L. Ahn, A. Bortz, and N. Hopper, "k-Anonymous Message Transmission," in *Conference on Computer and Communications Security,* pp. 122-130, 2003.

[14] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk, "Enhancing Source-Location Privacy in Sensor Network Routing," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems,* pp. 599-608, 2005.

[15] M. Shao, W. Hu, S. Zhu, G. Cao, S. Krishnamurth, and T. La Porta, "Cross-Layer Enhanced Source Location Privacy in Sensor Networks," in *IEEE SOCEOM '09,* pp. 1-9, 2009.

[16] J. Kong, and X. Hong, "ANODR: anonymous on demand routing with untraceable routes for mobile ad-hoc networks," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing,* pp. 291-302, 2003.

[17] X. Du, Y. Xiao, M. Guizani, and H. Chen, "An Effective Key Management Scheme for Heterogeneous Sensor Networks," in *Ad Hoc Networks,* vol. 5, issue 1, pp. 24-34, 2007.

[18] X. Du, "QoS Routing Based on Multi-class Nodes for Mobile Ad Hoc Networks," in *Ad Hoc Networks,* vol. 2, issue 3, pp. 241-254, 2004.

[19] M. Stephens, "EDF Statistics for Goodness of Fit and Some Comparisons," in *Journal of the American Statistical Association,* vol. 49, no. 268, pp. 730-737, 1974.

[20] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems,* pp. 162-175, 2004.

[21] "The Network Simulator - ns-2," http://www.isi.edu/nsnam/ns/, accessed August 31, 2009.

[22] C. Karlof, and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures," in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications,* pp. 113-127, 2003.

[23] M. Demirbas, and Y. Song, "An RSSI-based Scheme for Sybil Attack Detection in Wireless Sensor Networks," in *Proceedings of the 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks,* pp. 564-570, 2006.

[24] D. Loh, C. Cho, C. Tan, and R. Lee, "Identifying Unique Devices Through Wireless Fingerprinting," in *Proceedings of the First ACM Conference on Wireless Network Security,* pp. 46-55, 2008.

[25] J. Hall, M. Barbeau, and E. Kranakis, "Detection of Transient in Radio Frequency Fingerprinting Using Signal Phase," in *Wireless and Optical Communications*, ACTA Press, 2003.

[26] T. Kohno, A. Broido, and K. Claffy, "Remote Physical Device Fingerprinting," in *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93-108, 2005.

[27] D. Faria, and D. Cheriton, "Detecting Identity-Based Attacks in Wireless Networks Using Signalprints," in *Proceedings of the 5th ACM workshop on Wireless security,* pp. 43-52, 2006.

[28] K. Yedavalli, B. Krishnamachari, S. Ravula, and B. Srinivasn, "Ecolocation: a sequence based technique for RF localization in wireless sensor networks," in *Proceedings of the 4th international symposium on Information processing in sensor networks,* p. 38, 2005.

[29] X. Du, and F. Lin, "Maintaining Differentiated Coverage in Heterogeneous Sensor Networks," in *EURASIP Journal on Wireless Communications and Networking*, vol. 5, issue 4, pp. 565-572, 2005.

[30] "CRAWDAD: A Community Resource for Archiving Wireless Data at Dartmouth," http://crawdad.cs.dartmouth.edu/meta.php?name=rutgers/noise, accessed February 23, 2009.

[31] "ORBIT Emulator," http://www.orbit-lab.org/, accessed February 23, 2009.

[32] S. Zhong, L. Li, Y. Liu, and Y. Yang, "Privacy-Preserving Location-based Services for Mobile Users in Wireless Networks," Yale Computer Science, Tech. Rep. YALEU/DCS/TR-1297, July 2004.

[33] E. Elnahrawy, X. Li, and R. Martin, "The Limits of Localization Using Signal Strength: A Comparative Study," in *Proceedings of the First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks,* pp. 406-414, October 2004.