SHAH, PRATIK. Feasibility study of secure and robust location determination in current generation of wireless sensor networks. (Under the direction of Dr. Peng Ning).

Location Determination has been a fundamental requirement in many wireless sensor network applications. Various schemes have been proposed to solve this problem. These schemes depend on the measurement of physical quantities such as time of flight, angle of arrival, time difference of arrival and signal strength for location determination. Measurements in the real world are also affected by environmental conditions and contain unavoidable errors. Statistical techniques such as MMSE have been shown to be tolerant towards such errors. However in hostile environments, attackers can alter the measurements significantly to render these proposed schemes useless. Security mechanisms such as authentication and encryption can thwart external attacks such as eavesdropping and spoofing. However, attacks specific to location determination schemes differ from conventional security attacks and have been shown to be successful even when adequate security mechanisms are in place. Recently AR-MMSE, LMS and Voting-based schemes have been proposed to resist these attacks. A technique has also been proposed for detection of attacker nodes. This thesis presents the design and implementation of a nesC library that achieves secure and robust location determination using these techniques and provides a simple interface that can be used by high level applications. A working system was built using Cricket sensors to evaluate the feasibility of the techniques along with basic security mechanisms. We measure the tradeoffs between the time required for computation, memory consumption and the accuracy of the estimated location. We also measure the accuracy of the estimated location under various degrees of attack for both 2-dimensional and 3-dimensional scenarios. Our experimental results show that in a 2-dimensional system, even with 2 malicious Beacon Nodes out of 8, the maximum increase in error is less than 8 cm for all three techniques when the maximum error is 2 cm without any malicious Beacon Nodes. In case of 3-dimensional system with 1 malicious Beacon Node out of 8, the maximum increase is less than 20 cm with maximum error of about 10 cm when no malicious Beacon Nodes are present.
Feasibility study of secure and robust location determination in current generation of wireless sensor networks

by

Pratik Shah

A thesis submitted to the Graduate Faculty of North Carolina State University in partial satisfaction of the requirements for the Degree of Master of Science

Computer Science

Raleigh

2006

Approved By:

Dr. Ting Yu

Dr. Douglas S. Reeves

Dr. Peng Ning
Chair of Advisory Committee
To my parents...
Biography

Pratik Shah was born in Mumbai, India, in 1981. He received his Bachelor’s degree in Computer Science from Mumbai University, Mumbai, India in 2003. Since 2003, he has been a Masters student in the Department of Computer Science at North Carolina State University. In Summer and Fall of 2004 he worked at IBM Corp., RTP, NC, as a software development intern.
Acknowledgements

I acknowledge the efforts of Dr. Peng Ning, my thesis advisor, in providing me direction and guidance when required. He helped me remain focused and persistent. Working with him has been a pleasure and honor as well as a great learning experience. His dedication to work will be an inspiration for me throughout my life.

I am grateful to my committee members Dr. Douglas S. Reeves and Dr. Ting Yu for devoting time and providing valuable inputs.

Special thanks to An Liu for the valuable discussions related to implementation of this project.

I would also like to thank my roommates, Jaideep, Manav, Saurabh and Raj and all my pals in Raleigh for making my life at NCSU wonderful and interesting.
## Contents

### List of Figures

vi

### List of Tables

ix

### 1 Introduction

1.1 Background .................................................. 1
1.2 Motivation .................................................. 2
1.3 Contributions ............................................... 3
1.4 Thesis Organization ......................................... 4

### 2 Related Work

2.1 Range-based Localization Techniques ....................... 5
2.2 Range-free Localization Techniques ........................ 7
2.3 Secure and Robust Localization Techniques .................. 8
   2.3.1 SeRLoc ................................................. 8
   2.3.2 SPINE ............................................... 9
2.4 Sensor Security ............................................. 9
   2.4.1 TinyKeyMan .......................................... 10
   2.4.2 TinySec ............................................. 11

### 3 Background

3.1 Estimation Techniques ..................................... 12
   3.1.1 Minimum Mean Square Error Estimation ............... 12
   3.1.2 Attack Resistant MMSE ............................... 14
   3.1.3 Least Median Square .................................. 15
   3.1.4 Voting-based Location Estimation .................... 17
3.2 Malicious Node Detection .................................. 18
3.3 Cricket Nodes .............................................. 18

### 4 System Architecture

4.1 Non-Beacon Node ........................................... 22
4.2 Beacon Node ................................................ 23
List of Figures

3.1 Cricket Positioning Error .............................................. 19
3.2 Cricket Ultrasound Properties ........................................ 20
4.1 Before Deployment ...................................................... 21
4.2 After Deployment ....................................................... 21
4.3 Non-Beacon Component Diagram ................................. 23
4.4 Beacon Component Diagram .......................................... 24
5.1 Non-Beacon Modules Interaction .................................... 32
5.2 Beacon Modules Interaction ......................................... 32
5.3 TinyOS Packet .......................................................... 35
5.4 Beacon Advertisement .................................................. 35
5.5 Location Request Message ........................................... 35
5.6 Authentication Request ............................................... 35
5.7 Authentication Reply .................................................. 35
5.8 Location Message ....................................................... 35
5.9 Report Malicious Message ............................................ 36
5.10 Ultrasound Request .................................................... 36
5.11 Ultrasound Reply ....................................................... 36
6.1 3D Setup ................................................................. 38
6.2 2D Setup ................................................................. 38
6.3 Varying the Time with 8 Beacons and 1 Non-Beacon ............. 41
6.4 Varying the number of Non-Beacons for 5 Beacons .......... 41
6.5 2D Accuracy without Malicious Nodes ......................... 45
6.6 3D Accuracy without Malicious Nodes .......................... 45
6.7 2D Voting varying Granularity ..................................... 46
6.8 3D Voting varying Granularity ..................................... 46
6.9 2D LMS varying number of subsets .............................. 47
6.10 3D LMS varying number of subsets ............................. 48
6.11 2D varying number of Beacon Nodes ........................... 49
6.12 3D varying number of Beacon Nodes ........................... 50
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.13</td>
<td>2D with 1 Malicious</td>
<td>50</td>
</tr>
<tr>
<td>6.14</td>
<td>2D with 1 Malicious</td>
<td>51</td>
</tr>
<tr>
<td>6.15</td>
<td>2D with 1 Malicious</td>
<td>51</td>
</tr>
<tr>
<td>6.16</td>
<td>3D with 1 Malicious</td>
<td>52</td>
</tr>
<tr>
<td>6.17</td>
<td>3D with 1 Malicious</td>
<td>52</td>
</tr>
<tr>
<td>6.18</td>
<td>3D with 1 Malicious</td>
<td>53</td>
</tr>
<tr>
<td>6.19</td>
<td>2D with 2 Malicious</td>
<td>53</td>
</tr>
<tr>
<td>6.20</td>
<td>3D with 2 Malicious</td>
<td>54</td>
</tr>
<tr>
<td>6.21</td>
<td>3D with 2 Malicious</td>
<td>54</td>
</tr>
<tr>
<td>6.22</td>
<td>2D with 3 Malicious</td>
<td>55</td>
</tr>
<tr>
<td>6.23</td>
<td>2D with 3 Malicious</td>
<td>55</td>
</tr>
<tr>
<td>B.1</td>
<td>Base Station Interface Diagram</td>
<td>65</td>
</tr>
<tr>
<td>B.2</td>
<td>Non Beacon Interface Diagram</td>
<td>66</td>
</tr>
<tr>
<td>B.3</td>
<td>Beacon Interface Diagram</td>
<td>67</td>
</tr>
<tr>
<td>C.1</td>
<td>GUI Snapshot</td>
<td>69</td>
</tr>
</tbody>
</table>
## List of Tables

6.1 Number of messages for Various Operation ........................................... 40
6.2 Processing Time of Various Operation .................................................... 42
6.3 Library Memory Consumption ................................................................. 44
6.4 Estimation Memory Consumption ............................................................. 44
6.5 2D Voting Memory Consumption .............................................................. 47
6.6 3D Voting Memory Consumption .............................................................. 47
6.7 2D LMS Memory Consumption ................................................................. 48
6.8 3D LMS Memory Consumption ................................................................. 48
Chapter 1

Introduction

1.1 Background

Wireless sensor networks have experienced an explosive growth during the last few years. These tiny devices package together a circuit board with a microcontroller, application software, interfaces to sensors that can detect changes in temperature, pressure, moisture, light, sound or magnetism and a wireless radio that can report on their findings—all powered by a pair of AA batteries. These devices can potentially benefit a variety of scientific, military and commercial applications. They however present severe constraints in terms of available memory, processing power and energy. As a result most of the traditional computer applications are not suitable for them. Considerable research has been devoted to developing algorithms and protocols which work within these constrains to accomplish basic tasks such as communication, data aggregation as well as providing security for these devices.

The applications themselves are inherently data-centric, since individual sensors are not as important as the data collected by them. This data can seldom be meaningful if it is not accompanied with the location of the sensor. Suppose a sensor reports that the temperature is 30°C then that has little value unless coupled with the location of the sensor. Since wireless sensors networks are adhoc in nature, the location of the sensor cannot be fixed or known apriori. Thus each sensor has to determine its location dynamically.
One of the popular location determination (also called localization) system used in a wide variety of applications is the GPS. However, it is not suitable for use with the tiny sensors that are constrained by cost, memory, processing power and energy. GPS also does not work well in indoor environments. Hence a number of schemes have been proposed for addressing this issue [30, 21, 2, 6, 27, 22, 29]. These schemes require a few special sensors, called beacon nodes (also called anchor nodes) that have the capability of determining their locations. These beacon sensors then periodically broadcast a special signal, called beacon signal, that includes their locations. Sensors that need to determine their own locations receive these signals to measure their distances (or angles or received signal strength) from these beacons. Thereby, they can determine their own locations based on measurements from a number of beacons. This measurement often contains unavoidable noise. Estimation techniques are therefore employed for location determination.

Location determination system is also prone to security attacks in hostile environments like most other sensor network applications. They can be susceptible to conventional and non-conventional attacks. Some of the conventional attacks are wormhole attacks [10], Sybil attacks [20] as well as message replay attacks [12]. Many security mechanisms such as packet leashes, authentication and the use of nonces have been proposed to counter these conventional attacks. Examples of Non-conventional attacks that are possible are misleading location provided by the beacons as well as manipulation of the beacon signals that can cause enlargement or reduction of the measured distance [3, 33]. Such attacks cannot be prevented by conventional security techniques such as encryption and authentication alone. With the number of beacons and non-beacons deployed being large, it becomes difficult to prevent such attacks. Hence attack resistant techniques such as AR-MMSE, Voting-based scheme [5] and LMS [33] have been proposed. Liu et al. [4] propose a technique to detect wormhole and replay attacks using the beacon sensors.

1.2 Motivation

Many new and exciting applications have been developed which utilize the location of an entity to provide better and enhanced services. Generic platforms for location based services such as the Location Operating Reference Model (LORE) [32] have been proposed and developed. However, they are dependent on the underlying core location determination
system for proper functioning. This work has been motivated by the need to ascertain the plausibility of a robust and secure location determination system with the current generation of low cost wireless sensors. The developed system should provide a single well defined interface which can be used by higher level applications.

Real-world errors differ from the error model assumed in theory and the accuracy of the proposed techniques needs to be verified in such environments. Only with experimental results, can we justify the use of these techniques in practical systems. We also need to measure the resistance of these techniques in presence of error introduced by malicious Beacon Nodes.

1.3 Contributions

We make the following contributions in this thesis

1. This thesis describes the design and implementation of a nesC library, named TinyS-RLoc, that provides the current location of the sensor to an application running on the sensor itself, using an API. The library provides the following functionalities:
   - Secure and robust 2-dimensional location determination using AR-MMSE, Voting-based scheme [5] as well as LMS [33].
   - Secure and robust 3-dimensional location determination using AR-MMSE, Voting-based scheme [5] as well as LMS [33].
   - Detection of malicious Beacon Nodes [4].

2. This thesis describes the working of a secure and resilient location determination system in which the above library interacts with TinySec for cryptographic operations, TinyKeyMan for key establishment and management and the Cricket hardware for measurement purposes.

3. This thesis evaluates the feasibility of the location determination system using these robust location determination techniques in both 2-dimensional and 3-dimensional settings.

4. A working demonstration shows the effectiveness and robustness of the above system with a GUI, displaying results for various scenarios.
1.4 Thesis Organization

In Chapter 2, we review the related work in localization. In Chapter 3, we provide background information about the algorithms that are evaluated by the system as well as the hardware used. In Chapter 4, we explain the overview and details about the system, its different entities and software components. In Chapter 5, we explain the working of the TinySRLoc library. We present our experimental results in Chapter 6, followed by our conclusion and discussion of future work in Chapter 7.
Chapter 2

Related Work

As mentioned earlier, the first requirement for location determination is a measurement signal which can either be physical such as a radio signal, an ultrasound signal or an infrared signal or a quantity such as hop count. Hence, on the basis of the type of signal used, location determination schemes are categorized into range-based and range-free schemes. We describe some of the representative systems and schemes below. We also discuss SeRLoc and SPINE, which have been proposed for secure location determination. The techniques used in our system are different from these two since they are robust against malicious Beacon Nodes rather than trying to identify them.

An overview of the related work in the field of security for wireless sensor networks follows.

2.1 Range-based Localization Techniques

Range-based schemes use the measurement of the physical signal to determine the distance between the Beacon Node and the Non-Beacon Node. Various properties have been proposed for measurement such as Time of Arrival (ToA) of the signal, Angle of Arrival (AoA) and Time Difference of Arrival (TDoA).

- Cricket

The Cricket Location-Support System [27] was designed for an indoor environment
to provide location information using specially built Cricket sensor nodes. It was optimized for centimeter-level resolution, achieved through the precise placement of beacons on the ceiling and walls of each room. Each beacon transmits a string of data describing its space, which is delineated using short-range radio signals. A randomized algorithm allows multiple uncoordinated beacons to coexist in the same space. Since Cricket uses space identifiers rather than a co-ordinate system, it requires only one beacon per space. Cricket addresses the privacy of a determined location because the beacon and the other nodes (called listeners) do not communicate with each other and the listener determines its location based on the nearest beacon it hears. The measurement is done using TDoA technique, where each beacon transmits an ultrasound pulse along with the radio signal. The difference in time of the receipt of the two signals is used to estimate the distance. Cricket sensors are built using inexpensive components, costing less than $10 per sensor.

- APS using AOA

In [21], the authors propose the use of APS using angle-of-arrival (AOA) for localization. In this algorithm, nodes iteratively obtain position and orientation information from beacon nodes at the starting. The orientation is measured by determining the angle between the node’s own axis and direction of the received signal. The direction can be determined using an array of antennae or by using multiple ultrasound receivers. APS also utilizes the Distance Vector Routing concept, so that each node that has determined its location forwards it to its neighbors, to reduce the number of beacons required in the system. However, this introduces the problem of error propagation. Also, using angle-of-arrival is expensive and obtaining precise angle estimates is often difficult.

- AHLoS

AHLoS[29] uses ultrasound technology similar to Cricket. It uses a specially built Medusa node which has an Atmel ARM Thumb processor, four ultrasound transmitters and four receivers. Each node measures its distance to at least 3 beacons and performs maximum likelihood estimation using atomic multilateration. Once a node determines its location, it assumes the role of a beacon for other nodes to determine their locations. This is called iterative multilateration. In case, enough number of
beacons are not available, the authors propose collaborative multilateration where location information over multiple hops is utilized. However, this is computationally intensive. Hence a high beacon density is required.

2.2 Range-free Localization Techniques

- **APIT**
  The APIT method [30] isolates the environment into triangular regions between beacon nodes. It uses the point-in-triangle test for determining if the node is within a given triangle formed by a set of 3 beacons. When a sufficient number of triangles are obtained, it uses a grid algorithm to calculate the maximum area in which a node will likely reside. This is infeasible for sensor networks, hence each node maintains a beacon table, which is exchanged with each of its neighbors periodically and determines the triangles in which it lies using that information. This can only be achieved if the node density is high. APIT also assumes a larger radio range for beacon nodes.

- **DV-Hop**
  DV-Hop [22] is based on the distance vector routing algorithm. It assumes a heterogeneous network consisting of sensing nodes and beacons. First, beacons flood their locations throughout the network increasing the hop-count at each node. Nodes store the received beacons locations and the corresponding minimum hop-count from them. Secondly each beacon calculates the average hop size based on beacon signal from other landmarks and then broadcasts it to the nodes. Once the number of hops to at least three landmarks is known, nodes use the average hop size estimate to determine their distance to the landmarks and apply multilateration to determine their absolute location. Thus the communication cost of DV-Hop is very high when compared to other algorithms.

- **Centroid**
  In [2], Bulusu et al propose a very simplistic positioning system using an idealized radio model. The beacons are deployed in such a way that they form a mesh and have overlapping regions of coverage. A node uses the radio connectivity to a set of beacons to determine its coordinates. The coordinates are obtained as the centroid of
all the beacons in radio range. The position accuracy is about 30% of the separation distance between beacons, so the method needs a high beacon density to work well. The beacons must also be synchronized to prevent collisions.

- Convex Positioning

In [6], Doherty et al. present a scheme in which the position of each node is computed in a centralized manner, by collecting all node distances and by solving an optimization problem that minimizes the location error of each node. Nodes which can hear each other are constrained to lie within a certain distance of each other. This convex constraint problem is formulated as a linear programming problem, which is solved to find a globally optimal solution. The method requires centralized computation. For the technique to work well, beacon nodes need to be placed on the outer boundary, preferably at the corners. This gives tight constraints that yield a useful configuration. With 10% beacons, the error of locations of the nodes is on the order of the radio range.

2.3 Secure and Robust Localization Techniques

2.3.1 SeRLoc

In [16], the authors propose a secure range-free localization scheme that achieves decentralized passive localization. Each beacon is equipped with directional antennae, thus covering different sector areas with different transmissions. The beacon signals consist of the beacon location in terms of coordinates as well as the boundaries of the transmission sector. A sensor that needs to determine its location collects the locations of all the beacons in range and their advertised sectors. It maintains a grid score table to determine the overlap of all the sectors from heard beacons. It maintains a count for all the points in the grid and increments the count whenever the point lies in a beacon sector. The location of the sensor is computed as the center of gravity of all the points with highest count.

Using SeRLoc, sensors are able to detect attacks against wireless sensor networks such as the wormhole attack and the Sybil attack, with the assumption of no jamming of the wireless medium. In [15] the authors augment the role of the beacon by having one of the beacons verify the location estimated by the sensor using Verifiable Multilateration [3].
besides using active beacons as compared to SeRLoc, where the node request the beacons to transmit their location to determine its location. In addition, the node reports its computed location back to the beacons for verification. Although [15] requires low number of beacons, the beacons need to be equipped with directional antennae and must also be powerful enough to perform nanosecond processing and time measurements. The sensors themselves must also be able to perform nanosecond processing to provide distance verification.

2.3.2 SPINE

SPINE [3] is a range-dependent secure location determination scheme, which estimates the location of a node by verifying the distances of the node to at least three beacons using Verifiable Multilateration. The location estimation is performed centrally and once a node is aware of its location, it also becomes a beacon. Each beacon thus acts as a verifier. The task of the verifiers is to authenticate the node, estimate the distance to the node and report it to the central authority. The central authority computes the location of the node based on the reported distances from the verifiers and their locations using methods such as Minimum Mean Square Estimation. After this, it verifies if the measured distances lie within the measurement errors with estimated distances to each verifier. Next, it verifies if the computed location lies within at least one triangle formed by three verifiers. Once it passes both these tests, the computed location is stored and also communicated back to the node.

Though SPINE is robust against attacks in wireless sensor networks, it requires the initial deployment of a high number of reference points to achieve localization. Also the verifiers are required to measure time with a nanosecond precision for distance verification to the node, which can only be achieved using dedicated hardware.

2.4 Sensor Security

A number of different kinds of attacks specific to sensor networks have been identified. Attacks such as Sybil attacks, wormhole attacks, replay attacks and denial of service are just to name a few. As a result, security in sensor networks has attracted a lot of attention in the past several years. To counter the cryptographic attacks, encryption and
authentication solutions have been explored that are viable for the resource constrained sensors. TinySec \[11\] provides 80-bit SkipJack and RC5 symmetric cipher implementation for sensors. \[19\] suggests the use of Diffie-Hellman for establishing a key based on Elliptic Curve Discrete Logarithm Problem. This key can then be used with TinySec for encrypted communication. TEA \[31\] has also been proposed for encryption in sensor networks.

Various key pre-distribution techniques have also been proposed. Key Management has also been studied in \[17\]. To enable broadcast authentication, a protocol named \(\mu\)TESLA has been investigated \[25\]. A refinement to this is the multi-level \(\mu\)TESLA \[18\]. In \[10\] the authors introduce packet and geographical leashes to counter wormhole attacks. \[28\] discusses the idea of secure aggregation of information. Secure Time synchronization techniques have also been recently proposed by \[7, 13, 14\].

2.4.1 TinyKeyMan

TinyKeyMan provides the implementation of the key establishment technique based on a polynomial pool-based key predistribution proposed by \[17\]. In this technique, a pool of randomly generated bi-variate t-degree polynomials is used. If two sensors, say i and j, share the same polynomial, then they can establish a symmetric key by evaluating the polynomial at ID of i and j. These polynomials are distributed to the sensors before deployment. \[17\] discusses two ways to distribute these polynomials to the sensors viz. the random subset assignment scheme and the grid-based scheme. To reduce the computation at the sensors, each polynomial is evaluated at the sensor’s ID to give a polynomial share specific for that sensor, which is then loaded into the sensor. To determine if sensors i and j share the same polynomial, they first exchange their lists of polynomials. In case they do not share a polynomial then a neighboring sensor is contacted which shares a key with both i and j, to compute a random key for i and j to use. This key is called the path key. The authors show that the proposed schemes are quite resilient to compromised nodes. As long as t sensors are not compromised, an attacker cannot determine the other keys established with the same t-degree polynomial.
2.4.2 TinySec

TinySec [11] has been designed to provide link layer cryptography in wireless sensor networks and has been tightly integrated with TinyOS. It provides implementations of SkipJack and RC5 block ciphers on TinyOS platform. It provides two modes of operation for communication namely, Authenticated Encryption and Authentication only. Authentication only is the default mode of operation.

For encryption of messages which are larger than 64 bits, the default block size of the ciphers, it uses the ciphers in CBC mode. It uses a specially formatted 8 byte IV for the CBC mode. This IV is constructed from the headers in the message and a counter.

It uses the same block cipher encryption in CBC-MAC mode to generate a 4 byte Message Authentication Code (MAC) for each message. To provide additional security, it XORs the encryption of the message length with the first plaintext block.

TinySec by default relies on a single key manually programmed into the sensor before deployment. Although it can support multiple or pairwise keys, but TinySec must be re-initialized with the appropriate key before cryptographic operations can be performed. This becomes a problem when we cannot predict the sender before the receipt of the encrypted message.
Chapter 3

Background

In this chapter we present the techniques used in our system. We also describe the cricket sensor nodes used in our system and their limitations.

3.1 Estimation Techniques

The purpose of estimation is to reliably estimate the location of a sensor based on measured distances from the beacons, which are assumed to contain errors. The beacon location and the measured distance to it together are referred to as a location reference. The estimation technique is provided a number of such location references for location determination.

In our system we evaluate three robust estimation techniques, AR-MMSE, Voting based and LMS. AR-MMSE and LMS can be considered to be an enhancement to the basic MMSE technique. We describe these techniques below.

3.1.1 Minimum Mean Square Error Estimation

[29] describes a technique based on the Minimum Mean Square Estimation (MMSE) to estimate the location of the sensor as exactly as possible. MMSE utilizes the error between the estimated distance and the actual value of the distance as the basis for optimality. This error term is squared and minimized for the estimation. MMSE is also used in the
GPS system, with the difference being that Time of Arrival is used in GPS for measurement and Time Difference of Arrival used in [29]. The method as described in [29] works as follows

1. The location references are represented as an overdetermined system of equations. A system of equations in which the number of equations exceeds the number of unknowns is said to be overdetermined.

\[ F(x_0, y_0) = \sum_{i=1}^{k} \alpha_i^2 f(i)^2 \]

\[ f_i(x_0, y_0) = d_i - \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} \]

where \( k \) is the number of equations, \((x_0, y_0)\) is the estimated location of the sensor, \( \alpha_i \) represents the weight applied to each \( f_i \) and \( d_i \) is the measured distance between \((x_i, y_i)\) and \((x_0, y_0)\). Each \( \alpha_i \) is assigned as 1 in this case. Each \( f_i \) represents the error in the measured distance and is set to 0 for estimating \( x_0 \) and \( y_0 \) for the minimum error. Expanding and rearranging the terms, we get

\[-x_i^2 - y_i^2 + d_i^2 = (x_0^2 + y_0^2) + x_0(-2x_i) + y_0(-2y_i)\]

2. This system of equations is then linearized in terms of the unknown \( x_0 \) and \( y_0 \). The \((x_0^2 + y_0^2)\) term is eliminated from \( k \) such equations by subtracting the last equation from the rest. Thus we get

\[-x_i^2 - y_i^2 + x_k^2 + y_k^2 + d_i^2 - d_k^2 = 2x_0(x_k - x_i) + 2y_0(y_k - y_i)\]

which in matrix form is \( C = Ab \).
and
\[
b = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}
\]

This linearization is important since the calculations required to solve the non-linear form is prohibitively large and not viable for a sensor node. Although this may not give the optimum value but it will give a solution and within the smallest mean squared error.

3. This solution in terms of matrices is given by \( b = (A^T A)^{-1} A^T C \) as given by [9].

Since the system is overdetermined we need a minimum of 3 location references for a 2-dimensional system and 4 location references for a 3-dimensional system.

### 3.1.2 Attack Resistant MMSE

Attack Resistant MMSE (AR-MMSE) [5] is an enhancement to the basic MMSE scheme. The basis of AR-MMSE is that the mean square error of a set of location references is directly related to the inconsistency of the set. The mean square error (MSE) is mean of the square of error in the measured distance with each beacon and the distance calculated from the estimated location using MMSE method described above. A subset of the complete set of location references with the least MSE is hence the most consistent and is thus closest to the actual location of the sensor.

Computing the MSE for all subsets of the complete set is computationally intensive for the sensors. Hence a greedy approach is adopted. First the location is estimated using the complete set of location references and MSE is calculated. If it is below a threshold \( \tau \), then AR-MMSE returns the estimated location. Otherwise, the location is estimated from all subsets with one less location reference. If the least MSE is still higher than \( \tau \), the same method is repeated with all subsets of the set with the least MSE with one less location reference. This continues till a consistent set is encountered or the number of location references falls below the minimum required for estimation.

The intuition behind the algorithm is that the more inconsistent a set of location references is, the greater the corresponding mean square error should be. Thus if a malicious beacon node increases the location estimation error by introducing greater errors,
it correspondingly increases the mean square error of the set which contains that location reference at the same time.

Determination of the threshold $\tau$ is very important since if a lower value is chosen then we may never get an output and if a higher value is chosen then a malicious location reference could also become part of a consistent set. Theoretically if the number of location references is high such as 9 or greater, then the value of $\tau$ is chosen as 0.8 times the maximum distance error. Otherwise if the number of location references is low such as 4 or 5 then the value of $\tau$ is chosen as the 1.2 times the maximum distance error. This gives the probability close to 0.9 for the MSE of a consistent set to be less than the square of maximum distance error.

3.1.3 Least Median Square

[33] proposes the use of Least Median Square (LMS) for location estimation. The intuition behind using the median square is that outliers have little effect on the cost function, which in our case is the error between the estimated and measured distance (also called residue). LMS can tolerate up to 50% outliers in absence of noise [33]. A number of fixed size subsets from the complete set of location references are chosen, so that hopefully at least one subset does not contain outliers. This subset will have the least median square of the residues for all location references and hence will give the closest estimate to the real location of the sensor.

The procedure for implementing the robust LMS algorithm given by [33] is as follows:

1. A subset size $n$ is chosen based on the minimum number of location references required. For the 2-dimensional case, $n$ is chosen as 4 while for 3-dimensional case, $n$ is chosen as 5. The number of subsets, $M$ is then chosen based on the contamination ratio $\epsilon$, which is the estimated fraction of location references from malicious beacons, the required probability $P$ of getting at least one good subset without contamination and the subset size. The equation used for estimating $M$ is given by

$$P = 1 - (1 - (1 - \epsilon)^n)^M$$

Thus, larger the chosen $n$, smaller is the resulting $P$. 

2. M subsets of size n are then randomly drawn from the set of location references and an estimate $\theta_j$, for each subset is calculated using **Linear Least Squares** (LS).

3. Residues from each location reference in the complete set is then calculated for each $\theta_j$ and the median square $r^2_j$ is determined.

4. The least median square residue $r^2_m$ is found. The term $s_0$ is calculated based on $r^2_m$ as follows

$$s_0 = 1.4826(1 + \frac{5}{(N-p)})\sqrt{(r_m)^2}$$

where $p$ is the dimension of the estimate $\theta$. The term $(1 + \frac{5}{(N-p)})$ compensates for the tendency for a small scale estimate when there are few location references.

5. Based on this the weight $w_i$ calculated for each location reference $i$ is given by

$$w_i = \begin{cases} 1, & |\frac{r_i}{s_0}| \leq \gamma \\ 0, & otherwise \end{cases}$$

where $\gamma$ is a pre-determined threshold.

6. Then LS method is performed on the set of location references with $w_i = 1$ to get the final estimated location.

**LMS** differs from the AR-MMSE in three ways.

- It uses the least median square as the deciding factor.

- It uses random subsets of fixed size for estimation using LS method and thus does not improve upon the estimated location in the previous round. Least square minimizes the sum of squares of error as compared to mean of square of error in MMSE. Random subsets are chosen because computing the LMS from each possible subset is computationally prohibitive.

- While AR-MMSE calculates the MSE only for the subset, LMS calculates the median square error for the complete set of location references for each estimated location.
3.1.4 Voting-based Location Estimation

Voting based method [5] does not use statistical estimators such as LS or MMSE for estimation, instead it relies on basic geometry for estimation. The method works as follows

1. It first constructs the minimum bounding rectangle that covers all the beacon nodes, along with their transmission range, based on the location references.

2. It then divides this rectangle into a grid of square cells and stores a count variable for each of these cells. The number of cells is defined as the granularity and is taken as an input parameter.

3. The cells count variable is incremented by 1 if it falls in a beacon's transmission range.

4. The geometric centroid of all the cells with the maximum value of the variable is chosen as the nodes location.

In case of a 2-dimensional system, the beacon’s transmission range is modelled as a circular ring with inner radius as \(\max\{\delta - \epsilon, 0\}\) and outer radius \((\delta + \epsilon)\) where \(\epsilon\) is the measurement error and \(\delta\) is the measured distance. The minimum bounding rectangle is divided into 9 regions for each cell and with at most 6 comparisons, it can be determined in which region the beacon lies.

In the 3-dimensional system, the transmission range is modelled as a circular sphere and the region as a rectangular solid divided into cubes. In this case the bounding rectangular solid is divided into 27 different regions. The number of comparisons increases to 8 in this case.

The size of each cell is the tradeoff obtained between required precision and computing and memory overhead. To achieve a certain accuracy and for tradeoff between computation and accuracy, granularity is defined. If the achieved accuracy is less than the required accuracy, the voting based scheme is re-applied iteratively to the region of the cells with maximum count, in the previous iteration.
3.2 Malicious Node Detection

[4] describes a technique to detect malicious Beacon Nodes. In this technique a few beacons are programmed with a set of alternate IDs and their corresponding key material. Each of these beacons periodically assume the detection role and use one of these alternate IDs to act as a non-beacon node. They are termed as detecting beacons. They follow the same protocol as the non-beacon nodes. Thus a malicious beacon cannot distinguish between a detecting beacon and a valid non-beacon. After the detecting beacon has determined its distance with the neighbor beacon, it verifies its location by comparing the measured distance with the computed distance based on their neighbor’s broadcasted location and its own location. If there is a difference that is greater than the maximum allowable measurement error, but is within the maximum possible distance, then this is reported to the base station as a possible malicious beacon. In case the computed distance is greater than the maximum possible distance, the possibility of a wormhole exists and this is also reported to the base station.

To prevent malicious nodes from reporting against benign nodes, the base station maintains two counters for each of the beacons viz. alert counter and report counter. The alert counter maintains the number of alerts reported against a particular beacon while the report counter maintains the number of alerts a particular beacon has reported. The base station has been preprogrammed with two thresholds $\tau_1$ and $\tau_2$. For every report generated by a detecting beacon, the base station verifies if the report counter of the detecting beacon is below $\tau_2$ and the reported beacon has not been revoked. If yes, then it increments the respective counters of both the beacons. Once the alert counter for a particular beacon crosses the threshold $\tau_1$, it must be revoked. However if the report counter exceeded $\tau_2$ for the detecting beacon, the base station ignores the report.

3.3 Cricket Nodes

Cricket nodes [1] are based on Mica2 Hardware and have been designed and developed by a number of researchers at MIT Computer Science and Artificial Intelligence Laboratory. It has a pair of a ultrasound transmitter and a receiver. The transmitter emits short ultrasonic pulses of 40Khz frequency for 150\(\mu\)s. This pulse is detected at the receiver and a hardware interrupt is generated. For the TDoA technique, a radio signal and an
ultrasound pulse both are emitted simultaneously by the beacon. The non-beacon starts a
timer at the receipt of the radio signal and stops the timer at the receipt of the ultrasound
pulse. The measured time duration is the time taken for the ultrasound signal to travel
between the two nodes, since the radio signal travels at speed of light and hence its time
of flight can be negligible. The measured time duration, when multiplied with the speed of
sound gives the distance measurement.

However the accuracy of this distance measurement is dependent on a number of
factors. The first and foremost of all is the accurate value of speed of sound, which varies
with temperature, atmospheric pressure as well as humidity. Also, there has to be a clear
line of sight between the two nodes. Lastly, errors can be introduced by the hardware due
to activation and detection circuits or due to variable delays in notification of the interrupt
by TinyOS. Figure 3.1 from [26] shows the distance measurement error at different angles
and distances between the beacon and the non-beacon. As seen in Figure 3.2, the strength

![Figure 3.1: Cricket Positioning Error](image)

of the ultrasound signal from the beacon reduces considerably below -10dB and the signal
cannot be reliably detected at the non-beacon at larger distances, once the angle between
the perpendicularchs at the beacon and non-beacon crosses 80°.
Figure 3.2: Cricket Ultrasound Properties.  
Source: Manufacturer of US hardware for Cricket nodes, Kobitone Audio Company
Chapter 4

System Architecture

In this chapter we describe the entities in the system by providing a high level view of the software application running on them and the interaction of the software with interaction with TinySRLoc.

The system consists of the following entities:

- Non-Beacon Node
• Beacon Node
• Base Station
• Setup Server

As shown in the Figures 4.1 and 4.2, prior to deployment, the Beacon Nodes and the Non-Beacon Nodes are programmed by the Setup Server with the appropriate key materials and the application programs. After deployment, the Non-Beacon Node communicates with the Beacon Nodes and determines its location. This is communicated to the Base Station via an encrypted message. The Base Station upon request, also sends encrypted messages to the Beacon Nodes, instructing them of their locations. Now we describe the high level view of the software on each of these entities and their functions in detail.

4.1 Non-Beacon Node

We call the application program running on the Non-Beacon Node \(^1\), the *driving application*. The functions of the driving application on the Non-Beacon Node are implementation specific. However the driving application depends on our TinySRLoc library for getting the sensor’s location. The driving application can use the location information for its own purposes or instruct the library to report the current location to the Base Station. The library in turn depends on the driving application for providing it the beacon advertisements as it hears them. This is because the beacon advertisement might include information that is useful for the driving application itself or other applications but useless for the library.

For communicating with the driving application on the Non-Beacon node, TinySRLoc provides the NonBeacon module. This module manages the authentication of the Beacon Nodes and measurement of the distance to them. The driving application controls the time for which measurements must be done using start and stop commands. When the stop command is issued, estimation is performed by the module to determine the current location.

The NonBeacon module also informs the driving application of any malicious beacon encountered. Malicious Beacon Nodes are defined as Beacon Nodes that cannot

\(^1\)To avoid confusions, we refer to the physical sensor as a *Non-Beacon Node* and the software module running on it as *NonBeacon*. Similarly *Beacon Node* and *Beacon* are defined.
authenticate themselves, Beacon Nodes that have changed their location since their last authentication as well as ones which may be reported malicious by the Base Station. Each Non-Beacon Node shares a unique symmetric key with the Base Station for encrypted communication.

![Non-Beacon Component Diagram](image)

**Figure 4.3: Non-Beacon Component Diagram**

### 4.2 Beacon Node

The main function of the Beacon Node is to periodically transmit a beacon advertisement. This advertisement must at least contain the location of the Beacon Node. This location must be in terms of a pre-determined co-ordinate system. There can be a driving application on the Beacon Node as well. TinySRLoc provides the Beacon module for communication with the driving application. The Beacon module manages the periodic advertisement as well as transmitting the measurement signal upon request. Similar to the NonBeacon module, the driving application can control the functioning of Beacon module using start and stop commands.

For the Beacon Node, TinySRLoc also provides the Detection module which can be run after a certain time interval to detect malicious beacons in the neighborhood. This module communicates directly with the Base Station to report malicious Beacon Nodes. Similar to the Non-Beacon Node, each Beacon Node shares a unique symmetric key with the Base Station.
4.3 Base Station

The Base Station is generally assumed to be computationally powerful as compared to the Beacon and Non-Beacon Nodes. There can also be multiple Base Stations if each Beacon and Non-Beacon need to be in the radio range of the Base Station. The Base Stations could be organized in a hierarchical fashion. In our system, the Base Station stores the location of each Beacon Node and provides it to the Beacon Node upon request. Although this is not a requirement and the complete system can also function without the Base Station. The only required function of the Base Station is to inform the administrator of the reported and revoked malicious beacons. The Base Station also stores all the symmetric keys that it shares with the Beacon Nodes and the Non-Beacon Nodes. Hence, the Base Station must have adequate protection against tampering and capture. In our system, the Base Station is an ordinary sensor connected to a computer, via a serial port. The computer displays a graphical layout of the field along with the Non-Beacon Node’s reported locations.
4.4 Setup Server

The Setup Server programs the sensors with the appropriate application programs. It also generates the initial key material required for establishing keys between the Beacon Nodes and Non-Beacon Nodes. For this purpose, it needs to generate a random set of polynomials from a pool of polynomials for each sensor, based on parameters defined for key generation such as the degree of polynomial, number of polynomials per sensor and the total number of polynomials in the pool.

It also generates the pairwise key shared between the base station and each sensor. This key and the key material is also programmed into each of the sensors and the keys are then loaded into the Base Station software. Optionally we can also pass the Beacon Node locations to the Setup Server and those are also loaded into the Base Station software as well as the Beacon Nodes.

In case of detecting Beacon Nodes, the Setup Server generates a set of random IDs for them as well as their respective set of polynomials.

4.5 Assumptions

In this system we assume that the Non-Beacon Node is steady while measuring distances to Beacon Nodes. We also assume that the Beacon Nodes do not move from their location. This is because the Beacon Nodes cannot dynamically determine their location and are dependent on the Base Station to provide them with the location.

The more important assumption in the system is that the Base Station is not compromised until the time the Beacon Nodes can determine their location.

Although this is a simplistic approach, the system can be extended where the Beacon Nodes can initially assume the role of a Non-Beacon Node and determine their location based on the neighbors who know their locations with the assumption that all the neighbors are benign for at least that period of time initially.
Chapter 5

Implementation Details

5.1 TinySRLoc

TinySRLoc is a TinyOS [24] library written in nesC [8] that provides the secure and robust location determination. It acts like a central managing layer for the security, measurement and key establishment components on a sensor node. It is designed in such a way that it is independent of the way in which keys are managed and established, encryption and decryption are done, message authentication is done and verified as well as the way in which distance measurement is performed. Thus it is configurable based on hardware and software requirements. The main modules of TinySRLoc are

1. NonBeacon
2. KeyManagement
3. Measurement
4. SecPrimitive
5. Beacon
6. Detection

We now discuss the functioning of each of these modules along with design issues involved.
5.1.1 NonBeacon

The general protocol used in passive localization system is quite simple. Each Beacon Node periodically transmits a beacon signal along with a measurement signal when the channel is free. This is received at the Non-Beacon Node which performs the location estimation. Thus a Beacon Node does not communicate individually with any Non-Beacon Node or its neighboring Beacons. However to authenticate itself, the Beacon Node must communicate with each Non-Beacon. Hence we present our protocol.

1. Beacon advertisement - This is necessary since the Non-Beacon Node must hear from the Beacon Node to know its presence. The advertisement includes the Beacon Node’s location. We do not send an ultrasound pulse at this time.

2. The Non-Beacon Node on hearing a beacon advertisement, determines if the Beacon Node is a trusted one or not. If it has already been authenticated before, it sends a measurement signal request to it otherwise it sends a key establishment request.

3. Once the key is established, the Non-Beacon Node sends an authentication request, containing a random number, to the Beacon Node.

4. The Beacon Node replies to the authentication request by sending the same beacon signal appended with the random number and a MAC of the message computed using the established key.

The reason driving towards this design decision of the pull method rather than the basic push method is that when symmetric security is involved, unicast form of communication is required. The Beacon Nodes must have the appropriate key to authenticate themselves to the right Non-Beacon Nodes. Group keys cannot be used in this scenario since the Non-Beacon Nodes are expected to be mobile.

Decoupling of the measurement signal and the advertisement signal is also required since the system is designed to be as generic as possible. Although this does increase the energy consumed on the sensor since more communication is involved but the requirement of security demand it. There may be a requirement when every beacon signal needs to be authenticated. We provide a count for less stringent requirements where the Beacon Nodes need to be re-authenticated only after a certain number of beacon signals.
In case the key cannot be established with the beacon or if the authentication reply fails verification, the driving application is informed. The NonBeacon module maintains each Beacon Node information as a record in a fixed sized list. A valid records counter is also maintained for recently heard beacon signals because the authenticated beacon list is not erased. Upon successful estimation, the counter is reset. Whenever a beacon signal is repeated, the counter is incremented by one and the beacon record is moved to the position, indicated by the counter value, in the list. The counter is also reset whenever the distance measured with a Beacon Node changes by more than the measurement error from the stored information. This indicates that the Non-Beacon Node has moved from its original location. Since the memory of the sensor is limited, it only stores information for a fixed number of authenticated beacons. Once the list is full and new Beacon Node information needs to be added then it is added after the valid records in the list.

5.1.2 KeyManagement

Key establishment is provided by the KeyManagement module. This module provides a simple functions for establishing and looking up keys. Establishment of keys involves considerable processing and at least 2 radio transmissions. Hence the keys are stored in a list called the key list. When a key-establish request comes in, it first looks up the key list and tries to find the key for the specified sensor ID. If it exists, then it returns the key else it calls the underlying TinyKeyMan component to establish a key with the sensor. It generates an event whenever the key is established. The KeyManagement module manages the stored keys in a LRU fashion. When a previously established key is requested, it moves the key up in the list. Thus when the key list is full and a new key needs to be added, then it is added to the top of the list and the one at the bottom of the list is removed.

Another task of this module is to initialize the TinyKeyMan component with the assigned polynomials for the sensor. This is done whenever the module is initialized. In our system we use the random subset assignment technique for polynomial distribution to the sensors.
5.1.3 Measurement

At the Non-Beacon Node, the Measurement module is called at the time when the authentication request is sent as well as in case when the Beacon Node exists in the trusted list. This starts the ultrasound receiver. When the Beacon Node replies back to the authentication request, it sends a measurement signal along with it. To prevent overlap with replies from other Beacon Nodes, the module checks the sender ID in the message that it receives and signals an event when a valid measurement is recorded or a timeout occurs. Ideally the module, in case of a Non-Beacon Node, must detect the receipt of the first byte of incoming message of the correct type and immediately start the timer. However that is not possible since the type of the message cannot be determined by the first byte and there is also delay in starting or activating the timer. Hence the timer is started when the type of the message can be determined and compensation is added to the measured time to compensate for the previously received bytes of message as well as the delay in sending the signal by the Beacon Node. A compensation is also added for the timer offset, which is the time it takes for the timer to start. The measured time after adding the compensation values is multiplied with the speed of sound to determine the distance.

The CC1000 radio chip used on the Cricket also presents difficulties because the binary data does not arrive deterministically (the data does not arrive eight bits at a time). The offset of bits arriving late could represent the start of ultrasound. Thus at times the timer does not even start and measurement cannot be performed. So a workaround is used in the top level module, NonBeacon, where instead of waiting for the Measurement module to return the measurements before processing the next beacon, the measurements are updated independently whenever the Measurement module signals an interrupt. However it may happen that the Measurement module is instructed to measure the distance with another Beacon Node while it is waiting for the measurement signal from a previous Beacon Node. This is another reason why the different Beacon advertisements must be separated by at least 150ms, besides the reasoning and explanation provided in Section 6.2.

In case of the Beacon Node, the module is initialized when the Beacon module is initialized. It monitors the radio and detects the type of message for which an ultrasound signal needs to be sent. Again the type of the message can only be determined after a certain number of bytes are transmitted. This delays the transmission of the signal. There is additional delay in sending the signal due to actuation time of the hardware. However it
is compensated at the Non-Beacon Node.

5.1.4 SecPrimitive

The SecPrimitive module communicates with the ciphers provided by TinySec to provide encryption, decryption, MAC generation and MAC authentication. TinySec however has been tightly integrated with TinyOS to provide security at the link layer of the protocol stack. However TinySec requires manual key configuration and in our system we rely on pairwise keys established dynamically using TinyKeyMan. Hence we provide the security at the application layer using the SecPrimitive module.

This module uses the SkipJack cipher with 8 bytes block size and key size. We encrypt the messages between the sensors and the Base Station only, such as location request and malicious reports from the Beacon Nodes and the location information from the Non-Beacons.

One important consideration is the use of the Initialization Vector (IV). TinySec uses an IV to provide semantic security to the encrypted data. The IV is 8 bytes and computed using 4 bytes from the message header and 4 bytes generated using source address and a counter. Since we do not use the standard interface provided by TinySec, semantic security is provided by adding a random 16-bit value to the end of data and in case of malicious report after the sub-type field of the data. Thus the first block of plaintexts is different even when the original messages are same, resulting in a different and unrelated ciphertexts when used with CBC mode of encryption. As we see in the following section, we do not use the full 29 bytes of data portion provided in the TinyOS packet and thus can append the random number to our messages. Although the full 29 bytes of data are transmitted, the receiver can determine the size of each message based on its type, which is also encrypted, and discard the rest of the message.

5.1.5 Beacon

The Beacon module checks if it knows its location. If the location is not set, then it contacts the Base Station to retrieve its location. This request is also encrypted with the key shared with the base station.

Once it receives its location, it transmits the beacon signal periodically. Whenever
the Non-Beacon Node requests for an authentication message, it gets the key from the Key-
Management module and generates the MAC for the beacon message using that key. The
Measurement module on the Beacon, detects the type of message and sends an ultrasound
signal accordingly.

The Beacon Node does not store any information about the Non-Beacon Node except the established keys. Neither does it have knowledge about the neighboring Beacon Nodes.

5.1.6 Detection

The Detection module can be loaded into every Beacon Node or just a selected few. When the driving application issues a start to it, it randomly picks one of the detecting IDs from preloaded information and sets that as its system ID. It then loads the key material specific for that ID and re-initializes the KeyManagement module with that. Setting the system ID is important since TinyOS radio module discards any message that is not addressed to it except the broadcast messages. It then listens for beacon advertisements. On receiving one, it follows the same protocol as the Non-Beacon Node and verifies the measured distance with calculated distance. With the current hardware and measurement technique, we cannot implement the local replay detection as well as a wormhole detector as discussed in [4]. However we can still detect a Beacon Node whose calculated distance varies from the measured distance by more than the measurement error and report it to the base station. While the Detection module is started, the Beacon module is stopped and vice-versa so that they do not interfere with each other’s functions. Due to hardware constraints and large angles between the detecting Beacon Nodes and their neighbors, we set the maximum distance error to 20cm.

5.1.7 Modules Interaction

The interaction between the modules is shown in the Figures 5.1 and 5.2

For the NonBeacon

1. Receive Beacon Advertisement

2. Get/Establish Key
3. Key Established
4. Auth Request and Measure
5. Auth Reply
6. Measured Distance/Timeout
7. Estimate Location
8. Estimation Result
9. Encrypt Location
10. Send Location to Base Station

For the Beacon
1. Send Beacon Advertisement
2. Auth Request
3. Get Key
4. Key Established
5. Generate MAC
6. MAC
7. Auth Reply
5.2 Communication

TinyOS radio message structure does not include the sender address and neither does it support sub-types for messages. Hence we had to include these fields in the data portion of the message. Albeit this reduces the amount of data that can be sent in a message since the maximum data size is defined to be 29 bytes. The sender’s address is required in the system for identification. Since the most of the communication between a Beacon Node and Non-Beacon Node is unicast, sender’s address is important. It is required to establish a key between them, to authenticate the Beacon as well as to measure the distance to it. The sender address is a 16-bit number assigned by the administrator at the time of deployment of the sensors.

We used the type field in the message structure to differentiate between the types of communication, we define as channels. This reduces the complexity of message parsing and also provides the independence for each module to operate independently and receive its particular type of message directly. The basic channels are

1. Base communication Channel: These type of messages are only for base communication such as sending/receiving location information and malicious node information to the base station.

2. Key Setup Channel: These type of messages are used solely by the TinyKeyMan component for establishing keys.

3. Authentication Channel: These type of messages are used for sending an authentication request or receiving an authentication reply as well as for sending or requesting a measurement signal. This multiplexing works well since these signals are sent in the same order and the protocol is performed one Beacon Node at a time.

4. Plain Channel: This type of message is used by the Beacon Nodes for broadcasting their beacon advertisements.

TinyOS Generic Communication component allows the definition of different interrupt handlers for each of these channels. However care must be taken to prevent collisions. Thus before sending a message, a module needs to make sure that the different radio messages are not sent at the same time, thus overwriting the send buffer, before the message
has been successfully sent. Hence a flag is maintained which is set whenever a message is waiting for transmission and is cleared when the message is sent.

5.2.1 Packet Formats

Each of the channels has one or more sub-types. Shown below are the packet formats for each of them. The shaded portions are encrypted before sending. Figure 5.3 shows the default packet format in TinyOS.
Figure 5.3: TinyOS Packet

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dest ID</td>
<td>(2)</td>
</tr>
<tr>
<td>AM</td>
<td>(1)</td>
</tr>
<tr>
<td>Len</td>
<td>(1)</td>
</tr>
<tr>
<td>Grp</td>
<td>(1)</td>
</tr>
<tr>
<td>Data</td>
<td>(0-29)</td>
</tr>
<tr>
<td>CRC</td>
<td>(2)</td>
</tr>
</tbody>
</table>

Figure 5.4: Beacon Advertisement

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dest ID</td>
<td>(2)</td>
</tr>
<tr>
<td>Base Channel</td>
<td>(1)</td>
</tr>
<tr>
<td>Len</td>
<td>(1)</td>
</tr>
<tr>
<td>Grp</td>
<td>(1)</td>
</tr>
<tr>
<td>Src ID</td>
<td>(2)</td>
</tr>
<tr>
<td>REQUEST LOC</td>
<td>(1)</td>
</tr>
<tr>
<td>Random</td>
<td>(3)</td>
</tr>
<tr>
<td>CRC</td>
<td>(2)</td>
</tr>
</tbody>
</table>

Figure 5.5: Location Request Message

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dest ID</td>
<td>(2)</td>
</tr>
<tr>
<td>Auth Channel</td>
<td>(1)</td>
</tr>
<tr>
<td>Len</td>
<td>(1)</td>
</tr>
<tr>
<td>Grp</td>
<td>(1)</td>
</tr>
<tr>
<td>Src ID</td>
<td>(2)</td>
</tr>
<tr>
<td>AUTH REQUEST</td>
<td>(1)</td>
</tr>
<tr>
<td>Random</td>
<td>(2)</td>
</tr>
<tr>
<td>CRC</td>
<td>(2)</td>
</tr>
</tbody>
</table>

Figure 5.6: Authentication Request

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dest ID</td>
<td>(2)</td>
</tr>
<tr>
<td>Auth Channel</td>
<td>(1)</td>
</tr>
<tr>
<td>Len</td>
<td>(1)</td>
</tr>
<tr>
<td>Grp</td>
<td>(1)</td>
</tr>
<tr>
<td>Src ID</td>
<td>(2)</td>
</tr>
<tr>
<td>AUTH MSG</td>
<td>(1)</td>
</tr>
<tr>
<td>X</td>
<td>(2)</td>
</tr>
<tr>
<td>Y</td>
<td>(2)</td>
</tr>
<tr>
<td>Z</td>
<td>(2)</td>
</tr>
<tr>
<td>MAC</td>
<td>(2)</td>
</tr>
<tr>
<td>Random</td>
<td>(2)</td>
</tr>
<tr>
<td>CRC</td>
<td>(2)</td>
</tr>
</tbody>
</table>

Figure 5.7: Authentication Reply

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dest ID</td>
<td>(2)</td>
</tr>
<tr>
<td>Base Channel</td>
<td>(1)</td>
</tr>
<tr>
<td>Len</td>
<td>(1)</td>
</tr>
<tr>
<td>Grp</td>
<td>(1)</td>
</tr>
<tr>
<td>Src ID</td>
<td>(2)</td>
</tr>
<tr>
<td>LOC MSG</td>
<td>(1)</td>
</tr>
<tr>
<td>X</td>
<td>(2)</td>
</tr>
<tr>
<td>Y</td>
<td>(2)</td>
</tr>
<tr>
<td>Z</td>
<td>(2)</td>
</tr>
<tr>
<td>Random</td>
<td>(2)</td>
</tr>
<tr>
<td>CRC</td>
<td>(2)</td>
</tr>
</tbody>
</table>

Figure 5.8: Location Message
### Figure 5.9: Report Malicious Message

<table>
<thead>
<tr>
<th>Dest ID</th>
<th>Base Channel</th>
<th>Len</th>
<th>Grp</th>
<th>Src D</th>
<th>US: Mal</th>
<th>Count</th>
<th>Random</th>
<th>Na ID</th>
<th>Na ID</th>
<th>Na ID</th>
<th>US: Mal</th>
<th>Na ID</th>
<th>Na ID</th>
<th>Na ID</th>
<th>Na ID</th>
<th>Na ID</th>
<th>Na ID</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2)</td>
<td>(1)</td>
<td>(1)</td>
<td>(1)</td>
<td>(2)</td>
<td>(1)</td>
<td>(1)</td>
<td>(2)</td>
<td>(2)</td>
<td>(2)</td>
<td>(2)</td>
<td>(2)</td>
<td>(2)</td>
<td>(2)</td>
<td>(2)</td>
<td>(2)</td>
<td>(2)</td>
<td>(2)</td>
<td>(2)</td>
</tr>
</tbody>
</table>

### Figure 5.10: Ultrasound Request

<table>
<thead>
<tr>
<th>Dest ID</th>
<th>Auth Channel</th>
<th>Len</th>
<th>Grp</th>
<th>Src D</th>
<th>US: REQUEST</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2)</td>
<td>(1)</td>
<td>(1)</td>
<td>(1)</td>
<td>(2)</td>
<td>(1)</td>
<td>(2)</td>
</tr>
</tbody>
</table>

### Figure 5.11: Ultrasound Reply

<table>
<thead>
<tr>
<th>Dest ID</th>
<th>Auth Channel</th>
<th>Len</th>
<th>Grp</th>
<th>Src D</th>
<th>US: Signal</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2)</td>
<td>(1)</td>
<td>(1)</td>
<td>(1)</td>
<td>(2)</td>
<td>(1)</td>
<td>(2)</td>
</tr>
</tbody>
</table>
Chapter 6

System Evaluation

In this chapter we discuss two main aspects for evaluation of the system. The first is to set the optimum system parameters for maximum performance considering the tradeoffs with processing time, memory consumption and achieved accuracy and the second is to evaluate the performance of the attack resistant techniques with those parameters in terms of error in estimated location under various degrees of attack.

The most important parameters that affect the effectiveness of the system are:

- Beacon Interval
- Granularity for Voting-based scheme
- Number of Subsets for LMS
- Threshold values for AR-MMSE and LMS

Other parameters are implementation dependent. These include

- Memory allocated for location references and keys
- Maximum range of Beacon signal
- Number of polynomials per sensor and degree of each polynomial, the polynomial pool size and Auth count depending on security policy
- Detection Time Interval, $\tau_1$ and $\tau_2$ for Detection module
- Speed of sound value and correction values for Measurement module
- Block Size of the block ciphers for SecPrimitive module

6.1 Setup Description

The maximum range of the ultrasound signal goes down to about 4m for acceptable error of 5cm in measurements with angles up to 50° and we need at least 5 benign Beacon Nodes for estimation. So we placed 8 Beacon Nodes in a 3m x 3m x 2.5m area for evaluation of the system. Measurements were performed for 20 different locations with 10 readings at each location. 5 locations were chosen at the boundary of the area, approximately 50cm from the wall, where the measurement errors were larger than the ones at the center of the room due to larger angles between the Non-Beacon Nodes and the Beacon Nodes. The Beacon Nodes were placed in a regular arrangement with 4 Beacon Nodes on the ceiling and 4 on the walls as shown in Figure 6.1. All the 4 on the walls were placed at the same height and 2 had the same X coordinate and 2 had same Y coordinate. For the 2D case, we placed the Beacon Nodes 2 on each wall of the room in a regular arrangement as shown in Figure 6.2

![Figure 6.1: 3D Setup](image1)

![Figure 6.2: 2D Setup](image2)
6.2 Beacon Time Interval

The Beacon time interval refers to the time between successive beacon advertisements from a particular beacon. In a system like ours, where the Non-Beacon Nodes request for a measurement signal from the Beacon Nodes rather than the Beacon Nodes sending the measurement signal independently, the beacon advertisements must be spaced from each other such that the Non-Beacon Node completes the processing with the previous Beacon Node, before receiving the next advertisement. A TDMA based scheme would work very well in such a situation. But that would require tight synchronization among the Beacon Nodes and makes the design of the Beacon module more complex, which must now store information of neighboring Beacon Nodes.

We first calculate the channel utilization. The probability that a given Beacon Node is transmitting at a given instance is

\[ p = \frac{T_x}{I_e} \]

where \( T_x \) is the message duration and \( I_e \) is the expected interval between two transmissions. To accommodate 8 beacons, the least amount of time is 1200ms, when the processing time per beacon is 150ms. The probability of only one beacon transmitting at a particular time is

\[ P = p(1 - p)^{n-1} \]

where \( n \) is the number of beacons. The channel utilization is given by \( U = nP \) and the max utilization is achieved when \( p = \frac{1}{n} \) which is about 0.39 for 8 beacons. Thus we see that to guarantee that a Non-Beacon Node can process 8 beacons, it takes at least 3.2sec. To avoid collisions, randomized beacon advertisements as proposed in [26] are used. The Beacon Nodes sends a beacon advertisement at a random time between 600 and 1800 ms so that the expected interval is 1200ms.

Another scenario to be considered is when there are multiple Non-Beacon Nodes. Now when a Beacon Node sends a beacon signal, all the Non-Beacon receive it and all of them would try to contact the Beacon Node at more or less the same time and because of the collision detection only one of them can successfully communicate with the Beacon Node. Thus this will result in a competition for the Beacon measurement with each beacon advertisement and in turn increase the time a Non-Beacon requires for measuring distances to the minimum number of Beacon Nodes for estimation.
To evaluate this, we performed experiments for measuring the time taken for a single Non-Beacon Node to hear all 8 Beacon Nodes. We also measured the time in case where the number of Beacon Nodes is set to 5 and the number of Non-Beacon Nodes is varied. As expected, the average time to measure with all 5 Beacon Nodes increases with increase in the number of Non-Beacons. The average of 100 readings is shown in Figures 6.3 and 6.4.

The number of messages for each operation with each Beacon Node are shown in Table 6.1. The Beacon Advertisements is the number of advertisements in 4sec.

### 6.3 Processing Time and Energy Consumption

The energy constraints on the wireless sensors require that every implemented application be carefully designed to utilize the least possible time for processing. To evaluate the performance of our system for this criteria, we measured the time taken for each operation in the system. The radio operates at 19.2kbps and the size of the packet for TinyOS is 36 bytes, hence the travel time for each message is about 15ms. The average over 200 readings is provided in Table 6.2.

The Key establishment time is the time required to send an establish key request, receiving the reply, evaluating the common polynomial and updating the internal table. The Measurement time is the time taken to send a request, start the detector, receive the pulse and update the internal table. The Beacon Processing time is the time taken to process a beacon advertisement, lookup the internal table to verify if the Beacon Node has not changed the location and send the appropriate request. The Auth Verification time is the time required to lookup the key, verify the random number, verify the MAC and update

<table>
<thead>
<tr>
<th>Operations</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Establishment</td>
<td>at least 2</td>
</tr>
<tr>
<td>Measurement and Auth Verification</td>
<td>1</td>
</tr>
<tr>
<td>Beacon Advertisements</td>
<td>3</td>
</tr>
<tr>
<td>Auth Request</td>
<td>1</td>
</tr>
<tr>
<td>Measurement Request</td>
<td>1</td>
</tr>
<tr>
<td>Location Message to Base Station</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.1: Number of messages for Various Operation
Figure 6.3: Varying the Time with 8 Beacons and 1 Non-Beacon

Figure 6.4: Varying the number of Non-Beacons for 5 Beacons
<table>
<thead>
<tr>
<th>Operations</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Establishment</td>
<td>72.4</td>
</tr>
<tr>
<td>Key Lookup</td>
<td>0.1</td>
</tr>
<tr>
<td>Measurement</td>
<td>73.3</td>
</tr>
<tr>
<td>Beacon Processing</td>
<td>4.6</td>
</tr>
<tr>
<td>Auth Generation</td>
<td>1.2</td>
</tr>
<tr>
<td>Auth Verification</td>
<td>1.5</td>
</tr>
<tr>
<td>Encryption of Location Message</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 6.2: Processing Time of Various Operation

the internal table. Auth Generation time is the time taken to retrieve the key, generate the MAC and send the message.

The microcontroller on the Cricket sensors consumes 7mA of power when in active state and 10µA in the sleep mode. TinyOS maintains a queue and a scheduler based on FIFO for each task. Whenever the task list is empty, it puts the microcontroller to sleep, but keeps the other peripherals such as radio and timer on. An event handler is defined for each interrupt generated by these hardware peripherals and these define the tasks to be executed. So in our system, whenever the beacon timer expires, the microcontroller wakes up, sends a beacon signal and since there are no more tasks goes back to sleep. It then waits for 150ms for any Non-Beacon Node to reply. If there are no replies then even the radio is shut off to save more power and starts then again when the microcontroller wakes up.

The RF transmitter and receiver consume 7mA of power each when active and the transmitter consumes 0.1µA when idle, whereas the US transmitter consumes 50mA when active and 0.5µA when idle \[1\]. In our system, the microcontroller and RF transmitter on the Beacon Node is on for 150ms, whereas the US transmitter is on for 150µs between beacon advertisement intervals.

### 6.4 Memory overhead

The Memory overhead for each of the module as reported by TinyOS is provided in Table 6.3. The total memory consumption for the complete library, including the underlying software packages such as TinySec and TinyKeyMan is about 21k bytes of ROM and about
1.2k bytes of RAM in case of 2D and 24k bytes in case of 3D when the total memory available on the Cricket nodes is 128k bytes of ROM and 4k bytes of RAM. These values do not consider the Estimation module. The memory consumption of each of the techniques are given in Table 6.4.

We do not consider the overhead due to the key materials, since it is dependent on the implementation. Secondly, the key materials are loaded into the ROM, which can easily accommodate a number of polynomials with high degrees.

### 6.5 Estimation Evaluation

In this section, we present the evaluation of the performance of all the 3 robust estimation techniques compared to the basic MMSE scheme. First, we discuss how the various parameters were set based on experiments with 0 and 1 malicious Beacon Node. Following that, we measure the error in estimated location when there are 2 malicious Beacon Nodes. We also present the results in case of 2D scenario when there are 3 malicious Beacon Nodes.

For the three estimation techniques to be effective, their parameters must be set correctly. For Voting-based scheme, granularity must be set correctly. Higher granularity results in better accuracy of the estimated location. However higher granularity results in more number of comparisons (at most 8 per cell) as well as higher memory consumption. For LMS, the number of subsets and threshold are variable parameters. Lower threshold can result in rejection of a consistent subset and we may not get a result at all, while higher threshold can result in inclusion of the malicious location reference in the consistent subset. Also, higher the number of subsets, higher is the probability of getting a consistent subset. But with increased number of subsets, the memory consumption increases since the subsets need to be generated first and then LS estimation is performed on them individually. This also results in increased computation. For AR-MMSE, the value of the threshold is important for the same reasons as LMS. To estimate the correct parameters, we first need to evaluate the estimation error without any malicious Beacon Nodes. The estimation error in this case is due to the measurement error introduced by the hardware. We performed the measurements at the 20 locations. The average error using all the four methods are presented in Figures 6.5 and 6.6 for the 2D and 3D case respectively. The parameters set
Table 6.3: Library Memory Consumption

<table>
<thead>
<tr>
<th>Module</th>
<th>ROM (bytes)</th>
<th>RAM (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NonBeacon 2D</td>
<td>10860</td>
<td>823</td>
</tr>
<tr>
<td>NonBeacon 3D</td>
<td>13350</td>
<td>916</td>
</tr>
<tr>
<td>Beacon</td>
<td>15546</td>
<td>830</td>
</tr>
<tr>
<td>Base Station</td>
<td>22156</td>
<td>957</td>
</tr>
<tr>
<td>KeyManagement</td>
<td>4100</td>
<td>300</td>
</tr>
<tr>
<td>Measurement</td>
<td>1983</td>
<td>30</td>
</tr>
<tr>
<td>SecPrimitive</td>
<td>4030</td>
<td>61</td>
</tr>
<tr>
<td>Detection</td>
<td>4120</td>
<td>336</td>
</tr>
</tbody>
</table>

Table 6.4: Estimation Memory Consumption

<table>
<thead>
<tr>
<th>Module</th>
<th>ROM (bytes)</th>
<th>RAM (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMSE 2D</td>
<td>2462</td>
<td>52</td>
</tr>
<tr>
<td>AR-MMSE 2D</td>
<td>3734</td>
<td>110</td>
</tr>
<tr>
<td>Voting 2D</td>
<td>3642</td>
<td>100</td>
</tr>
<tr>
<td>LMS 2D</td>
<td>4164</td>
<td>408</td>
</tr>
<tr>
<td>MMSE 3D</td>
<td>3986</td>
<td>84</td>
</tr>
<tr>
<td>AR-MMSE 3D</td>
<td>5360</td>
<td>188</td>
</tr>
<tr>
<td>Voting 3D</td>
<td>9266</td>
<td>400</td>
</tr>
<tr>
<td>LMS 3D</td>
<td>6670</td>
<td>600</td>
</tr>
</tbody>
</table>

for this experiment were

2D case: Voting - Granularity : 100 , LMS - Number of Subsets : 6
3D case: Voting - Granularity : 400 , LMS - Number of Subsets : 10

We see that the average error is less than 20 cm in case of 3D when the number of Beacon Nodes is 6 or more and is less than 2 cm in case of 2D when the number of Beacon Nodes is 5 or more. Higher error in 3D is due to the higher error in measurement which is as high as 8 cm at times. The average error falls to about 10 cm for 8 Beacon Nodes.

Next, we measured the time required for computation and the memory consumption for Voting-based scheme and LMS. These are shown in Figures 6.7, 6.8, 6.9 and 6.10 and Tables 6.5, 6.6, 6.7 and 6.8. The processing time required for higher granularity in case of Voting-based scheme is very high. Hence we set the the granularity to 100 for 2D and 400 for 3D scenarios for the rest of the experiments. We also see that the memory required for larger number of subsets in case of LMS is very high. Hence we set that to 6 for 2D
Figure 6.5: 2D Accuracy without Malicious Nodes with Voting: Granularity=100, Required Accuracy=0 and LMS: Number of subsets=6

Figure 6.6: 3D Accuracy without Malicious Nodes with Voting: Granularity=400, Required Accuracy=0 and LMS: Number of subsets=10
Figure 6.7: 2D Voting varying Granularity

Figure 6.8: 3D Voting varying Granularity
Granularity | RAM (bytes)
---|---
20 | 20
40 | 40
60 | 60
80 | 80
100 | 100
120 | 120
140 | 140
160 | 160
180 | 180
200 | 200

Granularity | RAM (bytes)
---|---
100 | 100
200 | 200
300 | 300
400 | 400
500 | 500
600 | 600
700 | 700
800 | 800
900 | 900
1000 | 1000
1100 | 1100
1200 | 1200
1300 | 1300
1400 | 1400
1500 | 1500
1600 | 1600

Table 6.5: 2D Voting Memory Consumption  Table 6.6: 3D Voting Memory Consumption

Figure 6.9: 2D LMS varying number of subsets
Figure 6.10: 3D LMS varying number of subsets

Table 6.7: 2D LMS Memory Consumption

<table>
<thead>
<tr>
<th>Number of Subsets</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>48</td>
</tr>
<tr>
<td>3</td>
<td>144</td>
</tr>
<tr>
<td>5</td>
<td>240</td>
</tr>
<tr>
<td>7</td>
<td>336</td>
</tr>
<tr>
<td>9</td>
<td>432</td>
</tr>
<tr>
<td>11</td>
<td>528</td>
</tr>
<tr>
<td>13</td>
<td>624</td>
</tr>
<tr>
<td>15</td>
<td>720</td>
</tr>
<tr>
<td>17</td>
<td>816</td>
</tr>
<tr>
<td>19</td>
<td>912</td>
</tr>
<tr>
<td>21</td>
<td>1008</td>
</tr>
<tr>
<td>23</td>
<td>1104</td>
</tr>
<tr>
<td>25</td>
<td>1200</td>
</tr>
</tbody>
</table>

Table 6.8: 3D LMS Memory Consumption
and 10 for 3D scenarios. For 2 malicious Beacon Nodes out of 8, this gives a probability of about 90% for selecting a consistent set in both scenarios for LMS.

We also measured the average execution time by varying the number of Beacon Nodes. The results are shown in Figures 6.11 and 6.12. We see that varying the number of Beacon Nodes does not show as high variation as compared to varying the Granularity in Voting-based scheme and the Number of Subsets in LMS.

![2D Average Execution Time](image)

**Figure 6.11: 2D varying number of Beacon Nodes**

Having set the parameters at acceptable levels, we next measured the accuracy of location estimation by introducing one beacon as malicious. We programmed the malicious Beacon Node to increase its distance to the Non-Beacon in steps of 20cm and advertise its location accordingly. Thus the malicious Beacon Node tries to be farther from its real location.

We see that all the three techniques show very high accuracy, close to the case when there is no malicious beacon at all, in case of high location error introduced by the malicious Beacon Node. It is when the location error is small, that the threshold values are really tested. The number of Beacon Nodes was set to 7 in case of 3D and 6 in case of 2D. We set the required accuracy of Voting to 0 for both 2D and 3D.

Based on the results, we set the threshold for AR-MMSE to 6 and the threshold for LMS to 1.5.

Next, we see the results when 2 and 3 beacons are malicious. We set the number
Figure 6.12: 3D varying number of Beacon Nodes

Figure 6.13: 2D with 1 Malicious
Figure 6.14: 2D with 1 Malicious

Figure 6.15: 2D with 1 Malicious
Figure 6.16: 3D with 1 Malicious

Figure 6.17: 3D with 1 Malicious
Figure 6.18: 3D with 1 Malicious

Figure 6.19: 2D with 2 Malicious
Figure 6.20: 3D with 2 Malicious

Figure 6.21: 3D with 2 Malicious
Figure 6.22: 2D with 3 Malicious

Figure 6.23: 2D with 3 Malicious
of beacons to 8 for these experiments. We placed the Non-Beacon at a fixed position and first programmed 2 beacons on the ceiling to increase their distance to the Non-Beacon in steps of 20cm each in positive x, y and z direction and then programmed 3 beacons.

We see that the error in the estimated location for 2 malicious Beacon Nodes in the 2D case is less than 10cm for all the three techniques. Although Voting does show higher error when the error introduced by the malicious Beacon Node is up to 40cm, but still the maximum error in the estimated location is about 20cm. In case of 3 malicious Beacon Nodes, the error in estimated location varies considerably because the number of benign Beacon Nodes is just 5. Voting-based scheme does not show as high variation as the other 2 schemes, but the estimation error is still very high and variable and cannot justify the effectiveness of the schemes.

In case of 3D with 2 malicious Beacon Nodes, only LMS shows a consistent behavior. This is because we set the number of subsets with the assumption of 2 malicious Beacon Nodes. LMS does show higher error at low errors (upto 80cm) introduced by malicious Beacon Nodes. The other 2 schemes do not perform very well and show high variations.

Thus we can conclude that the schemes are effective with increase in error of estimated location of less than 8cm when there are 1 and 2 malicious Beacon Nodes in case of 2D.

Similarly all the 3 schemes are resistant to 1 malicious Beacon Node out of 8 in 3D scenario and show increase in error of less than 10cm with Voting-based scheme being the exception. Voting shows increase in error of about 20cm. When there are 2 malicious Beacon Nodes LMS does show high resistance.
Chapter 7

Conclusion

We had two major goals, first was to verify if, with the current generation of wireless sensor networks, a secure and robust location determination system was indeed feasible. For this purpose we used the Cricket nodes, which have distance measurement capabilities. These however have many limitations such as smaller angle is required between the sensors, clear line of sight between the sensor is needed, maximum distance between the sensors cannot be more than 4m and larger angles or distances, which are common in realistic scenarios, result in higher errors in measurement. Thus a better and more reliable means for distance measurement is required for the system to be practical. We built a demo system which works within these constraints and performed experiments to verify if the software is feasible. The software is feasible even with the current capabilities of the sensor networks in terms of memory capacity, processing power and available energy. We built the software such that the top level module is generic and whenever better measurement methods are available, only the measurement module needs to modified. We also see that the configuration and time required for deployment is low since all that is needed is to provide the Beacon Nodes with their accurate locations and proper key material. However higher density of beacons is required again due to limitations of hardware in terms of maximum range.

The next goal was to verify the performance of the three estimation techniques in terms of memory consumption, processing time and accuracy under various degrees of attack, both in case of 2-dimensional and 3-dimensional scenarios. We see that tradeoffs are
needed with the required accuracy and memory consumption or time consumption when setting the parameters for Voting-based scheme and LMS. We also see that with low degrees of attack, such as upto 2 malicious Beacon Nodes in case of 2D and upto 1 malicious Beacon Node out of 8 Beacon Nodes in case of 3D, the attack resistant techniques are very effective and show high resistance to the error introduced by the malicious Beacon Nodes. For higher degrees of attack, higher number of benign Beacon Nodes are needed.

7.1 Future Work

With the current generation of wireless sensor networks, the system is feasible but has limitations imposed by the hardware constraints. For example, the distance measurement error is high at higher angles and it also requires clear line of sight between the Beacon Nodes and Non-Beacon Nodes. With improved hardware and more advanced techniques for distance measurement, it needs to be seen whether this limitation could be removed and that too without increasing the energy consumption. Also if the ultrasound signals could be modulated to carry information, then the number of messages could be reduced and it would also allow detection of local replay and wormhole attacks.

Also in this work we haven’t considered that a Non-Beacon Node is compromised and reports its location wrongly to the base station. Two techniques have been proposed for verifying this situation viz. Verifiable Multilateration[3] and Secure Verification of Location Claims[23]. This can be implemented on the Base Station to which the Non-Beacon Node reports or the Beacon Nodes in direct range of the Non-Beacon Node. Also when a detecting Beacon Node detects a misbehaving node, there is a possibility that the reporter node is a benign node and the beacon signal is being locally replayed. Detection of this kind of attack using TDoA technique is very difficult. A suitable revocation scheme is also needed to limit the effect of the malicious Beacon Nodes. Augmenting these techniques to the developed system would definitely improve its scope and security.
Bibliography


[23] Secure Verification of Location Claims. N. sastry and u. shankar and d. wagner. ACM Workshop on Wireless Security (WiSe 2003), September, 2003. 7.1


[27] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. in ACM MOBICOM, August 2000. 1.1, 2.1


Appendix A

System Interfaces Details

1. NonBeaconNodeC
   Interface StdControl
   Interface: NonBeaconNode
   1. command result_t add_beacon(LocMsg* beacondata)
   2. command void purge_beacon_list()
   3. command result_t send_to_base()
   4. event result_t malicious_detected(uint16_t bid)
   5. event result_t loc_determined(Location* est_loc)

2. BeaconC
   Interface StdControl
   Interface: Beacon
   1. command result_t request_location()

3. Estimation
   Interface: Estimation
   1. command result_t estimate(Record *record, uint8_t num, Location *location, uint32_t threshold)

This interface is provided by modules MMSE_2DM, MMSE_3DM, ARMSE_2DM, ARMSE_3DM, Voting_2DM, Voting_3DM, LMS_2DM, and LMS_3DM.
4. DetectMaliciousC

  Interface: Detect
  1. command result_t detect()
  2. command result_t stop()
  3. command void process_auth(TOS_MsgPtr msgP)

5. KeyManagementC

  Interface: KeyManage
  1. command void init(uint16_t ID);
  2. command void getBaseKey(uint8_t* baseKey);
  3. command void getKey(uint16_t id);
  4. command void deleteKey(uint16_t id);
  5. command result_t lookupKey(uint16_t id,uint8_t* bkey);
  6. event result_t key_established(uint16_t id);

6. MeasureC

  Interface: Measure
  1. command void init()
  2. command result_t measure(uint16_t beacon_id)
  3. async event result_t measured_distance(uint16_t est_distance, uint16_t beacon_id, result_t rst)
Appendix B

Interface Diagrams

Figure B.1: Base Station Interface Diagram
Figure B.2: Non Beacon Interface Diagram
Figure B.3: Beacon Interface Diagram
Appendix C

GUI Snapshot

The computer that is connected to the Base Station provides a GUI, which plots the reported location of the Non-Beacon. It also provides functionality to inform the Beacon of their locations. The GUI has been programmed using Java SWT and communicates directly with the serial port of the computer. The communication with the Base Station is done using pre-defined message formats.