

Secure Localization for Wireless Sensor Networks

Ajay Kulhari, Binjia Jiao and Lingxuan Hu
Department of Computer Science
University of Virginia
Charlottesville, VA
[ak7q, bj3r, lh5g]@cs.virginia.edu

Abstract

Security in localization is a primary requirement due to strong dependence on location awareness by many protocols in the sensor networks. Prior research in sensor networks has studied localization problems in a non-adversarial setting assuming a trusted environment. In this paper we present the security issues in the DV-Hop localization algorithm and present protocols that can be applied to make the service efficient against most common attacks. We design a new algorithm grid counter that works better than multilateration to compute the nodes' positions and prevent position corruption against seed compromise. The security protocols presented are lightweight and efficient, using only highly efficient symmetric cryptographic primitives. The study and design of security protocols presented are the first set of security mechanisms provided in the localization area. Protocol analysis and simulation experiments are performed and the results are provided.

1. Introduction

Wireless sensor networks are emerging technologies that have a wide range of potential applications such as battlefield surveillance and emergency response [7]. Research on sensor networks generally assumes a trusted environment, but in many sensor network applications, the network will most likely be deployed in situations where an adversary may be motivated to disrupt the function of the network. An adversary may be able to position several non-legitimate nodes within the network and use them to transmit false messages. Further, an adversary may compromise a node in the network and gain access to its key material.

Location awareness is widely recognized as an important part of sensor networks. For example, location based routing protocol can save significant energy by eliminating the cost of route discovery. The localization done using globally accessible beacons such as GPS [6] is expensive and not consistent with the smaller and cheaper trend of sensor devices. Recently many localization

algorithms have been proposed to estimate per-node location information relying on several beacon nodes with known positions. However, the security properties of the localization algorithms have not been studied yet.

Secure localization is important since security breaches leading to miscalculation of position can drastically affect the proper functioning of WSNs. For example, in geographic routing, if a node gets its' location wrong, it will not be able to correctly forward messages to the next hop while using greedy geographic routing strategy. Many other applications such as tracking mobile targets, power management schemes etc. also require position awareness.

In this paper we study the possible attacks to current localized algorithms, specifically targeting at DV-Hop algorithm [1]. We implement security features, so that under the proposed trust model our protocol is able to provide proper localization to the nodes.

We make two contributions to the area of secure localization for wireless sensor networks. Firstly, we give a model for the type of attacks likely to happen in such a system. Secondly, we present the first set of security protocols in localization services. Lastly, our grid counter algorithm is a robust algorithm compared to multilateration used by DV-Hop for position calculation. The grid counter algorithm incorporates security at design level and work against seed compromise attacks. The protocols presented, combined with broadcast authentication, can authenticate seed nodes using the shared secret between the nodes and seeds. We also discuss in the paper the use of μ TESLA (an efficient broadcast authentication scheme that requires loose time synchronization) in providing authentication in our protocols.

In Section 2 of this paper, we discuss the related work in localization algorithms and security. We formalize the problem and present the threat model outlining major types of attacks in Section 3. In Section 4, we describe our trust model and assumptions about the network, and security schemes. Section 5 presents the protocols & countermeasures against the attacks and in Section 6 we do the evaluation and provide the simulation results.

2. Related Work

Many localization algorithms have been proposed for sensor networks [1, 2, 3, 4, 8, 9, 10]. The approaches to solve the localization problem in these algorithms differ in the assumptions of the network and the hardware capabilities.

Generally, localization algorithms can be classified as range-based and range-free solutions. Range-based solution uses distance estimates or angle estimates for location calculation. Several different techniques such as Time of Arrival (TOA), Received Signal Strength Indicator (RSSI), Time Difference of Arrival (TDOA), and Angle of Arrival (AOA) can be used in different environments [6, 9, 4, 10]. Range free solution has no assumption about the ranging hardware. In [8] a node uses the centroid of neighboring nodes' positions to obtain its location estimate. DV-HOP [1] and amorphous position algorithm [2] assumes a reasonably uniform and dense network. The coordinates of seeds are flooded throughout the network so that each node can get a hop-count to that seed. Nodes calculate their position based on the received seed locations and corresponding hop count. The details of the algorithm will be discussed in Section 3. APIT [3] is an area based approach by isolating the environment into triangular regions between beaconing nodes. Because of the hardware limitations of WSN devices, range free localization algorithms are cost-effective alternative to more expensive range based approaches.

Security mechanisms using asymmetric algorithms are likely to demand more computation and power consumption than what is feasible for many sensor network applications. μ TESLA [12] is a protocol for efficient, authenticated broadcast and flooding that uses only symmetric key cryptography and requires minimal packet overhead. μ TESLA achieves asymmetry necessary for authenticated broadcast and flooding by using delayed key disclosure and one way key chains constructed with a publicly computable cryptographically secure hash function. We use μ TESLA protocol for the authentication of seed nodes' coordinates. More details are presented in Section 5.

In [14] some attacks to routing protocols in sensor networks are proposed. One of the most severe and challenging attacks is wormhole attack, in which an attacker forwards messages through a high quality out-of-band link and replays those messages at another location in the network. A wormhole can easily create a sinkhole that attracts (but does not forward) packets to many destinations. Wormhole can also disrupt the localization process, as will be discussed in Section 3. One possible way to counteract wormhole attack is to use secure neighbor discovery protocols. Two techniques of secure neighbor discovery are presented in [11] and [12]

respectively. We assume either is used such that each node knows its proper neighbor set.

Security issues related to localization have not been discussed much. The only work that has been done is in [5] by Sastry and Wagner. In that paper, the authors tried to verify the position as advertised by the node to grant some resource based on its location. In our paper, we are focusing on more basic aspects of security in localization services, where the nodes should be able to calculate their positions correctly in presence of security threats.

3. Problem Statement

Before diving into specific security protocols, it helps to have a clear statement of the localization security problem. In the following sub-sections we target at the DV-Hop algorithm, outlining how it works, propose models for different classes of adversaries, and consider security goals in this setting.

3.1. DV-Hop algorithm

DV-Hop is an efficient localization algorithm for uniform and reasonably dense network. It doesn't need any ranging techniques, and has minimal requirement on the number of seed nodes (≥ 3).

The DV-Hop algorithm works as follows. A seed sensor initiates a gradient flooding by sending its neighbors a message with its location and a count set to one. Each recipient remembers the value of the count and forwards the message to its neighbors with the count incremented by one. Hence a wave of messages is propagated outward from the seed. Each sensor maintains the minimum hop count value received and ignores messages containing larger values, which prevents the wave from traveling backwards. If two sensors can communicate with each other directly (i.e. without forwarding the message through other sensors) then they are considered to be within one communication hop of each other. The minimum hop count value that a sensor maintains will eventually be the length of the shortest path to the seed in communication hops. Hence a gradient is essentially a breadth-first-search tree. Figure 1 shows the gradients propagating from one seed.

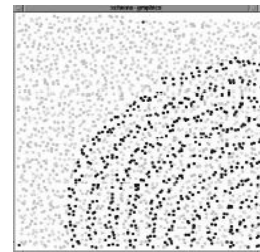


Figure 1: Gradients propagating from a seed.

Once a node can calculate the distance estimate to more than 3 seeds in the plane, it uses triangulation (multilateration) to estimate its location. In particular, each sensor estimates its coordinates by finding coordinates that minimize the total squared error between calculated distances and estimated distances. Theoretically, if errors exist in the distance estimation, the more seeds a node can hear the more precise localization will be.

Sensor j 's calculated distance to seed i is:

$$d_{ji} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

and sensor j 's total error is

$$E_j = \sum_{i=1}^n \left(d_{ji} - \hat{d}_{ji} \right)^2$$

Where n is the number of seed nodes and \hat{d}_{ji} is the estimated distance computed through gradient propagation. Least squared error coordinates can be found using gradient descent. The partial derivatives are:

$$\frac{\partial E_j}{\partial x_j} = \sum (x_j - x_i) \left(1 - \frac{\hat{d}_{ji}}{d_{ji}} \right)$$

$$\text{and } \frac{\partial E_j}{\partial y_j} = \sum (y_j - y_i) \left(1 - \frac{\hat{d}_{ji}}{d_{ji}} \right)$$

and incremental coordinate updates are

$$\Delta x_j = -\alpha \frac{\partial E_j}{\partial x_j} \text{ and } \Delta y_j = -\alpha \frac{\partial E_j}{\partial y_j}$$

where $0 < \alpha \ll 1$

The correction is the estimated distance per hop which is calculated by each seed as in described in [1].

3.2. Threat Model

We use a four hops path as an example to discuss attacks. The normal DV-Hop operation is shown in Figure 2. Seed node S broadcasts its coordinate and hop count 1 in plain text, other nodes will increase the hop count by 1 and rebroadcast the message until it is flooded to the whole network.

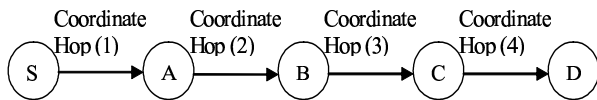


Figure 2: Normal DV-Hop Operation.

Possible attacks to this localization process are discussed in the following subsections. The attacks are classified as outsider attacks and insider attacks.

3.2.1. Outsider attacks. An outsider is a node that is not a member of the network and does not have key material to access the network. In DV-Hop algorithm, if messages are neither encrypted nor authenticated, an outsider can arbitrarily eavesdrop or forge messages in intermediate steps in flooding.

Message Forging: Since the purpose of secure localization is to ensure the integrity of location information, we think message confidentiality is not a serious concern here. But an outsider can impersonate seed node and broadcast bogus coordinates into the network, as shown in Figure 3. Other nodes will believe its authenticity and propagate the message to the whole network.

Preventing outsider from injecting bogus data into the localization process is very essential to guard the miscalculation of positions by the nodes. Some authentication scheme need to be incorporated into the protocol so that nodes can verify the identity of the seeds.

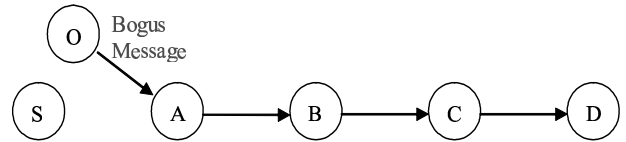


Figure 3: Outsider attack, Message Forging. Outsider node O broadcasts a bogus coordinate into the network.

Replay attacks: Some outsider replays valid messages from one part of network to other part. The original message may contain a smaller hop count value which may not be valid in the other part of network, as shown in Figure 4.

In the example, node D will receive hop count of 2 instead of the expected 4. Since node only uses minimal received hop count to estimate the distance, this attack successfully misleads the calculated distance to 1/2 of the true value.

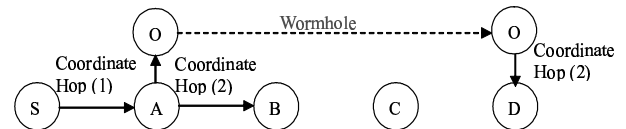


Figure 4: Outsider attack, Replay Attack. Two outside nodes O form a wormhole. They eavesdrop the transmission of A , tunnel and replay the message at D through the wormhole.

3.2.2. Insider attack. An insider who obtains key material from a node is more dangerous. Since the sensor devices are inexpensive with no tamper-resistant hardware, it is likely that a competent adversary with physical access to a device would be able to obtain the key material stored on the device. In addition to what an outsider can do, an insider can also forge valid messages. Notice that there are two kinds of nodes in the network (seed nodes knowing their positions and other nodes with unknown positions). Whether seed nodes or other nodes are compromised will put different threats on the localization process.

Compromised Seeds: If a seed node is compromised, it can broadcast a false coordinate to the network. It's hard to detect anomaly since only the seed node itself knows the true coordinate. The impact of this attack on final calculated locations of nodes can be significant if the false coordinates of the seeds is largely deviated from the true coordinates.

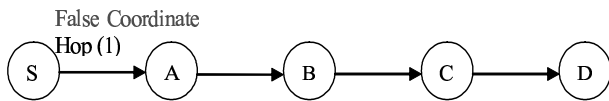


Figure 5: Insider attack, Compromised Seeds. Seed node S initiates a gradient with a false coordinate.

Compromised Nodes: The intermediate nodes can broadcast false hop count. They can either increase or decrease the hop count to disrupt the hop information, as shown in node A of Figure 6. The intermediate nodes can also choose not to forward the message, as shown in node C of Figure 6.

The impact of this attack needs to be carefully investigated. If the node decreases the hop count, the receiver will probably accept the count value since it always chooses the minimal received hop count to estimate distance. So the attack will usually succeed, and the impact on calculated location will be determined by how much the false hop value deviates from the true hop value.

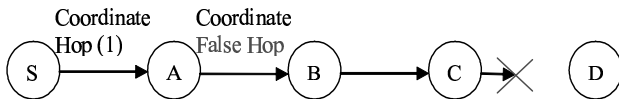


Figure 6: Insider attack, Compromised Nodes. Compromised node A sends a false hop count. Compromised node C discards the message to thwart the propagation of message.

However, if the node increases the hop count or just discard the message, the impact will be trivial. This is because in reasonably dense and uniform network, there are typically multiple paths to a node. In other words, a node will receive multiple messages, so one discarded

message can't prevent the message from propagating. The increased hop count will not be used by node since it only chooses the minimal hop count.

3.3. Security Goals

Our design is aiming at incorporating lightweight security mechanisms into DV-Hop algorithm to effectively prevent localization misbehavior. We are not concerned with the confidentiality of message transmission, but make sure that a node only uses the valid coordinates and hop count to calculate its location.

4. Trust Model and Assumptions

We make the following assumptions on the network.

1. *The network is reasonably dense and uniform.*
This is in fact the prerequisite for DV-Hop algorithm. We assume there are multiple paths from seed node to other nodes in the network and one disconnected path will not affect the overall performance.
2. *Nodes share symmetric key with each seed node.*
Each node in the network shares a symmetric key with seed nodes. Typically the fraction of seed nodes in the network is small, so it's reasonable for each node to share a key with each seed node. The shared key is used to authenticate the identity of seed node.
3. *Only a small fraction of nodes can be compromised.*
We assume that within the localization period it is not possible to compromise many nodes. The majority of nodes are not compromised.
4. *There is no involvement of base station.*
The base station is not required to participate in localization process.
5. *Each node already has its proper neighbor set.*
Several works [11, 12] have been proposed for secure neighbor discovery. We assume each node already has the proper neighbor set so the replay attack can be easily detected.

5. Countermeasures

5.1. Overview of μ TESLA

In this paper, we use μ TESLA [12] to authenticate the seed nodes. μ TESLA has been proposed for broadcast authentication in distributed sensor networks. μ TESLA employs a chain of authentication keys linked to each other by a pseudo random function, which is by definition a one way function. Each key in the key chain is the image of the next key under the pseudo random function. The

efficiency of μ TESLA is based on the fact that once a sensor node has an authenticated key in a key chain, only pseudo random function operations are needed to authenticate the subsequent broadcast messages. μ TESLA introduced asymmetry by delaying the disclosure of symmetric keys. A sender broadcasts a message with a Message Authentication Code (MAC) generated with a secret key K , which will be disclosed after a certain period of time. When a receiver receives this message, if it can ensure that the packet was sent before the key was disclosed, the receiver can buffer this packet and authenticate it when it receives the corresponding disclosed key. To continuously authenticate the broadcast packets, μ TESLA divides the time period for broadcasting into multiple time intervals, assigning different keys to different time intervals. All packets broadcasted in a particular time interval are authenticated with the same key assigned to that time interval.

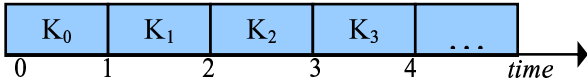


Figure 7: μ TESLA key chain. The keys are disclosed at different time intervals.

To authenticate the broadcast messages, a receiver first authenticates the disclosed keys. μ TESLA uses a one-way key chain for this purpose. The sender selects a random value K_n as the last key for the key chain and repeatedly performs a pseudo random function F to compute all the other keys: $K_i = F(K_{i+1})$; $0 \leq i \leq n-1$, where the secret key K_i is assigned to the i th time interval. With the pseudo random function F , given K_j in the key chain, anybody can compute all the previous keys K_i ; $0 \leq i \leq j$, but nobody can compute any of the later keys K_i ; $j+1 \leq i \leq n$. Thus, with the knowledge of the initial key K_0 , the receiver can authenticate any key in the key chain by merely performing pseudo random function operations.

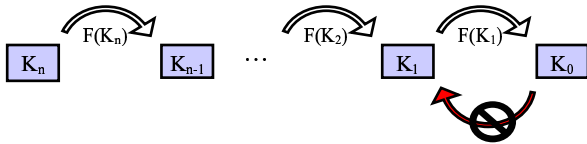


Figure 8: One way pseudo random function F . When K_i is disclosed, $K_0..K_{i-1}$ can be calculated through F , but no information about K_{i+1} can be achieved.

When a broadcast message is available in i th time interval, the sender generates MAC for this message with a key derived from K_i and then broadcasts this message along with its MAC and discloses the key K_{i-d} assigned to the time interval $i-d$, where d is the disclosure lag of the authentication keys. The sender prefers a long delay in order to make sure that all or most of the receivers can receive its broadcast messages. But, for the receiver, a

long delay could result in high storage overhead to buffer the messages. Each key in the key chain will be disclosed after some delay. As a result, the attacker can forge a broadcast packet by using the disclosed key.

But, μ TESLA uses a security condition to prevent a receiver from accepting any broadcast packet authenticated with a disclosed key. When a receiver receives an incoming broadcast packet in time interval I_i , it checks the security condition $[(T_c + \Delta - T_0)/T_{int}] < I_i + d$, where T_c is the local time when the packet is received, T_0 is the start time of the time interval 0, T_{int} is the duration of each time interval, and Δ is the maximum clock difference between the sender and itself. If the security condition is satisfied, i.e., the sender has not disclosed the key K_i yet, the receiver accepts this packet. Otherwise, the receiver simply drops it. When the receiver receives the disclosed key K_i , it can authenticate it with a previously received key K_j by checking whether $K_j = F^{i-j}(K_i)$, and then authenticate the buffered packets that were sent during time interval I_i .

5.2. Notations and Key Deployment

We use the following notations to describe security protocols and cryptographic operations:

- S_i is the i th seed
- K_{Si} denotes the secret MAC key shared between seed i and other nodes.
- $MAC_{K_{Si}}(M)$ denotes the computation of message authentication code (MAC) of message with the MAC key K_{Si} .

The initial key commitment is given to each node during startup. It can also be given by unicast message transmission to each node by seeds before localization starts. The second method is very costly so we propose to use the key deployment at startup. We specify the steps for key deployment and initialization.

1. All the nodes are initialized with key commitment K_0 , time(T_{now}) to sync with seed nodes, disclosure delay (d), starting time of interval i and interval duration (T_{int}) used for broadcast authentication
2. The one way pseudorandom function (F) used to generate the key chain is public and known to all the nodes.
3. The initial key chain with the 2 keys is generated and stored at the seed nodes. The interval is defined in which the seed nodes disclose their key.

5.3. Protocol and Analysis

We aim for resilience against outsider and insider attacks. The probability that correct seed coordinate information is transferred in the network degrades, when

more seeds or nodes are compromised. Our goal is to design simple and efficient mechanisms achieving high attack robustness.

We need an authentication mechanism with low computation and communication overhead. An inefficient authentication mechanism could be exploited by an attacker to perform denial of service attack by flooding nodes with malicious messages, overwhelming them with the cost of verifying authentication. Thus, for identity authentication of a message, we use a message authentication code (MAC) and shared key between seeds and other nodes.

We propose change in the message to account for authentication and node-to-node message integrity. We transform the hop count to node-ID list and hash-chain and add MAC of seed coordinate and seed ID for authentication. Now we analyze how various attacks are counteracted by our protocols.

(a) **Outside Attack:** The adversary node may try to behave as a seed and try to broadcast false positions to disrupt the correct calculation of position by the normal nodes. Here we will present a protocol based on the μ TESLA protocol proposed by Perrig et.al in [12] to authenticate the seed nodes. In the original version of μ TESLA the base station used to store a long key chain to avail the authenticated broadcast for many transmissions. But since the seed coordinate broadcast (hop count flooding) is done just once (assuming that the nodes are stationary and not mobile) for the nodes to compute their positions, we amend the protocol for the seed nodes to hold just two keys. The first key (K_0) serves as the key commitment while the second key (K_1) is used to encrypt the broadcast message carrying the seed coordinate.

The message format to be broadcasted by the seed is $M = [\text{Coord}(X_{si}, Y_{si}) \parallel \text{Seed-ID} \parallel \text{MAC}\{\text{Coord}(X_{si}, Y_{si}) \parallel \text{Seed-ID}\}_{K_{si}} \parallel \text{Hopcount}]$

Each node stores the message in the buffer till the key is disclosed. In the time-interval when the key is disclosed, each node acquires the key K_1 and checks if K_1 has been disclosed or not. If the key is newly disclosed it performs the check: $K_0 = F(K_1)$. If the check is unsuccessful the message is deleted from the buffer. Otherwise the node calculate the MAC on the $\{\text{Coord}(X_{si}, Y_{si}) \parallel \text{Seed-ID}\}$ using the key disclosed and if MAC matches then the packet is accepted and used to calculate the position.

(b) **Replay Attack:** We assume that each node keeps a list of all its immediate neighbors. This helps in detecting if the message received has come from an immediate neighbor. It is possible that some strong adversary replays a message to disrupt the correct hop count value in the data packet. But since each node possesses a list of its neighbors it can check if the message it has received has

come from its neighbor or not. A node thus accepts only those packets that are sent by its neighbors and discard the others.

(c) **Compromised Seeds:** If seed nodes are compromised it can affect the localization process by broadcasting wrong coordinates to the nodes. Since the key is known to the compromised node, it will be able to make itself authenticated. But the effect of the compromised seeds can be abated by checking the consistency of the received nodes.

The multilateration algorithm in DV-hop scheme doesn't provide consistency checking. It assumes all information is true and only tries to minimize the total estimate error.

We use a Grid Counter algorithm for consistency checking. Once the individual node gets ranging information from seed nodes, it aggregates the results through a Grid Counter algorithm. In this algorithm, a grid array is used to represent the maximum area in which a node is most likely to reside. In our experiments, the length of a grid side is set to $0.05r$, to guarantee that estimation accuracy is not significantly affected. We also allow the ranging to offset by at some distance since the estimated distance from hop count may not be accurate.

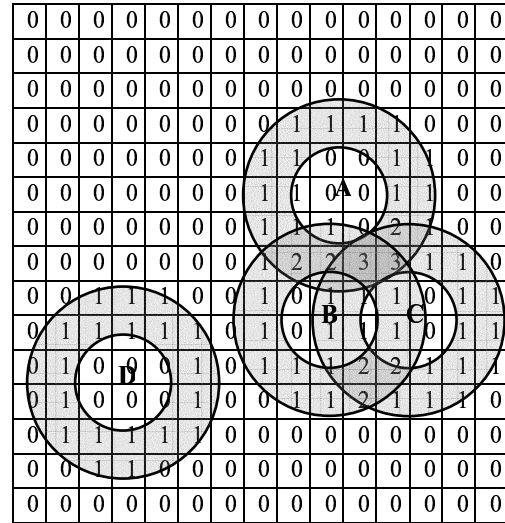


Figure 9: Grid count algorithm

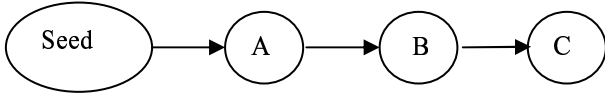
Then the advertisement of one wrong coordinate will be overwhelmed by the other true coordinates. Figure 9 shows an example. There are four seed nodes (A, B, C and D) in the network. The shaded region of each seed indicates possible regions of the node based on estimated distance (it's not a circle line but a torus because there may be some ranging errors). Assume D is compromised and advertises a wrong coordinate. The effect is some wrong locations are marked 1. But this count value is overwhelmed by the intersection of other three seed nodes

A, B and C (grid count = 3). So the compromised seeds do not mislead the calculated position when the fraction of compromised nodes is small. This is also verified by the experimental results.

(d) **Compromised nodes:** There can be normal nodes which can be compromised and the adversary has access to the key material of the node. The compromised nodes can affect the hop count value carried in the packet. The hop count is used to estimate the distance of the node from seeds. When the hop count is increased by the compromised node, it will not affect the position calculation significantly. The reason is that while calculating the position the nodes take into account the shortest path to the seed nodes.

When the hop count is decreased by a compromised node, it can affect the position significantly. We do not want the compromised node to decrement the hop count in the message. We modified the DV-Hop algorithm to carry the hop count value in a different way. Now instead of carrying a hop count value, the packet will carry the node IDs of the intermediate nodes. The node IDs are always appended in the message in the order of forwarding. The destination node counts the number of node IDs in the message and the total number reflects the hop count distance from the seed node. The security is provided by per-hop hashing. In this approach, each node forwarding the packet will append its ID to the message along with the hash value calculated as $H\text{-Chain} = H\{(ID) \parallel H\text{-chain}\}$ as shown in Figure 10. The hash function is a one way function and it provides integrity to the message by preventing the inversion of one-way hash function by compromised node.

Now the message propagated by the nodes looks like:
 $M = [\text{Coord}(X_{S_i}, Y_{S_i}) \parallel \text{Seed-ID} \parallel \text{MAC}\{\text{Coord}(X_{S_i}, Y_{S_i}) \parallel \text{Seed-ID}\}_{K_{Sij}} \parallel \text{ID-list} \parallel \text{Hash-chain}]$
 where, S_i is the i th seed, j implies the j th interval and K_{Sij} denotes the key used by seed i in the j th interval.



Seed:

$$\begin{aligned}
 H_0: & H(\text{ID}_{\text{SEED}}) \\
 \text{ID-List:} & \text{ID}_{\text{SEED}} \\
 M = & [\text{Coord}(X_{\text{SEED}}, Y_{\text{SEED}}) \parallel \text{ID}_{\text{SEED}} \parallel \\
 & \text{MAC}\{\text{Coord}(X_{\text{SEED}}, Y_{\text{SEED}}) \parallel \text{ID}_{\text{SEED}}\}_{K_{\text{SEED}j}} \parallel \text{ID-List} \parallel H_0]
 \end{aligned}$$

A:

$$\begin{aligned}
 H_1: & H(\text{ID}_A \parallel H_0) \\
 \text{ID-List:} & \text{ID}_{\text{SEED}} \parallel \text{ID}_A \\
 M = & [\text{Coord}(X_{\text{SEED}}, Y_{\text{SEED}}) \parallel \text{ID}_{\text{SEED}} \parallel \\
 & \text{MAC}\{\text{Coord}(X_{\text{SEED}}, Y_{\text{SEED}}) \parallel \text{ID}_{\text{SEED}}\}_{K_{\text{SEED}j}} \parallel \text{ID-List} \parallel H_1]
 \end{aligned}$$

B:

$$\begin{aligned}
 H_2: & H(\text{ID}_B \parallel H_1) \\
 \text{ID-List:} & \text{ID}_{\text{SEED}} \parallel \text{ID}_A \parallel \text{ID}_B \\
 M = & [\text{Coord}(X_{\text{SEED}}, Y_{\text{SEED}}) \parallel \text{ID}_{\text{SEED}} \parallel \\
 & \text{MAC}\{\text{Coord}(X_{\text{SEED}}, Y_{\text{SEED}}) \parallel \text{ID}_{\text{SEED}}\}_{K_{\text{SEED}j}} \parallel \text{ID-List} \parallel H_2]
 \end{aligned}$$

C:

$$\begin{aligned}
 H_3: & H(\text{ID}_C \parallel H_2) \\
 \text{ID-List:} & \text{ID}_{\text{SEED}} \parallel \text{ID}_A \parallel \text{ID}_B \parallel \text{ID}_C \\
 M = & [\text{Coord}(X_{\text{SEED}}, Y_{\text{SEED}}) \parallel \text{ID}_{\text{SEED}} \parallel \\
 & \text{MAC}\{\text{Coord}(X_{\text{SEED}}, Y_{\text{SEED}}) \parallel \text{ID}_{\text{SEED}}\}_{K_{\text{SEED}j}} \parallel \text{ID-List} \parallel H_3]
 \end{aligned}$$

Figure 10: Message propagation process. The seed is sending its coordinate and hop count to node C

We assumed that each node is aware of its neighbors. So when a node receives a packet, it checks for the outermost node ID to be its neighbor. This prevents a compromised node to forward the message without adding its id to node-list. The node then recalculates the hash on the node-ID list and matches it with hash-chain in the message. If they are equal then the node accepts the packet. If the compromised node removes some ID from Node-ID list to decrease the hop-count, the hash does not match and so the node discards the packet.

It is still possible for the compromised nodes to disrupt the packet by adding many node IDs in the message. But as we previously mentioned, if the hop count is increased by the adversary, it's most likely that the destination node will discard it.

(e) **Message Dropping:** It is also possible that compromised nodes will start dropping the packets. But since the network density is large and we assume that the number of compromised nodes is limited, there will be several paths to reach the nodes. So even if the compromised nodes decide to drop the packet, it will not affect the system much.

6. Evaluation

We simulated the DV-Hop localization algorithm using GloMoSim [15] with 40 nodes uniformly distributed in the network as shown in Figure 11. We randomly chose 19 of them as seeds which render a seed fraction of 48%. This fraction is relatively high, since we need a better input for multilateration. The more entries of seed coordinates we have for multilateration, the closer the results will be to real coordinates. The network dimension is 500m*500m, and radio range of sensors is 140.508m. We chose a small scale network with low density due to the limitation of GloMoSim and vast time consumed in multilateration process.

According to DV-Hop algorithm, we simulated two floodings, one is hop count flooding, and the other is

correction flooding. Each seed initiate a hop count flooding to its one hop neighbors, encapsulating its coordinates into the packet. It also adds some header information such as flooding type into the packet. When a node receives such a packet, it adds one hop to the packet and broadcasts it to its one hop neighbors. In this way, seeds' coordinates and hop count information is flooded across the whole network. Considering the collision issues, we made each node send out a packet after a small random time to avoid collision. Because nodes are unreliable, each seed floods a hop count packet twice to ensure better coverage.

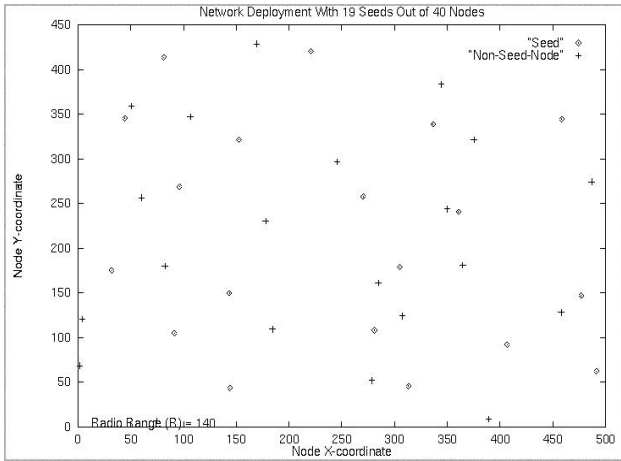


Figure 11: Network Deployment

The other type of flooding is the correction flooding. A correction is actually the estimated distance per hop. It is initiated after a seed has received enough coordinates and hop count information from other seeds. The number of received coordinates is controlled by a threshold parameter in our simulation.

From the simulation results, we observed that on an average the *correction (physical distance per hop)* is a bit over half of the radio range, which is compatible with what is described in [1] and [2].

Mathematical package Maple™ has been used to implement atomic multilateration process introduced in [2] and [4]. From the results we find that as an approximation technique, multilateration produce better results (close to real coordinates) when more seeds' coordinates are available. However, as a tradeoff, when more seeds' coordinates are available, multilateration process takes much more time to finish.

Through simulation and multilateration results, we illustrated the error in positioning caused by the four attack models: (1) A percentage of seeds are compromised, and they produce false coordinates; (2) A certain number of nodes maliciously decrease hop count in the packet and forward it on; (3) A certain number of nodes maliciously increase the hop count in the packet and forward it on; (4)

A certain number of nodes maliciously drop flooding packets. The position errors here refer to the difference of positions calculated under attacks from positions calculated under normal situations. The position errors are normalized by radio range.

It is illustrated by the position errors, that the compromised seeds attack and decrease hop attack are more severe (impacting more significantly on position accuracy).

We implemented the Grid Counter algorithm using Java. The performance of Grid Counter Algorithm and normal DV-Hop algorithm is compared under the compromised seeds attack.

In all the figures (Fig.12-15), node IDs in the horizontal axis represent those that are not seeds and need to use the algorithm to calculate their coordinates. There are totally 21 such nodes in the network.

6.1. Seed Compromise Attack

In the same simulation setting as a normal DV-Hop algorithm, we randomly chose 25% of the seeds as compromised seeds. These compromised seeds propagate $(2x, 2y)$ in the coordinate flooding packet instead of their real coordinate (x, y) . The comparison between the position error of the multilateration algorithm and the Grid Counter algorithm is shown in the Figure 12. The position error is normalized by radio range R .

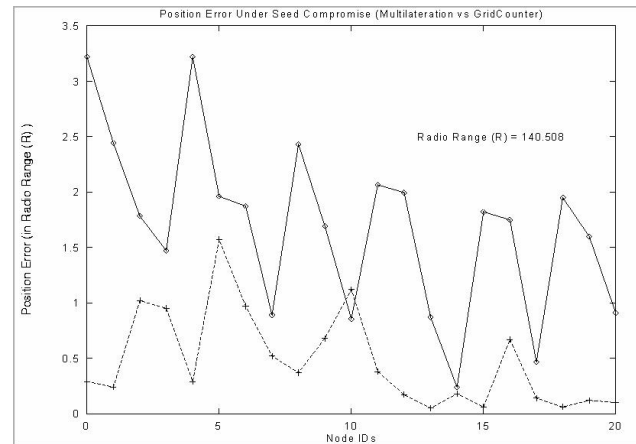


Figure 12: Multilateration vs Grid Counter Algorithm under Seed Compromise Attack

From Figure 12, we can observe that under seed compromise attack, Grid Counter algorithm works much better than multilateration process. Averaging position errors over all nodes, produces $1.69R$ error rate for multilateration and only $0.47R$ for Grid Counter algorithm. The reason is that multilateration takes every seeds' coordinate into account without considering security, while Grid Counter algorithm can filter out compromised seeds by selecting the grid with maximum hits.

6.2. Hop Count Increase Attack

Based on the setting of normal DV-Hop Algorithm, we randomly chose 10% nodes maliciously increasing the hop count by two each time instead of one while forwarding packet. The position error caused by this type of attack is shown in Figure 13.

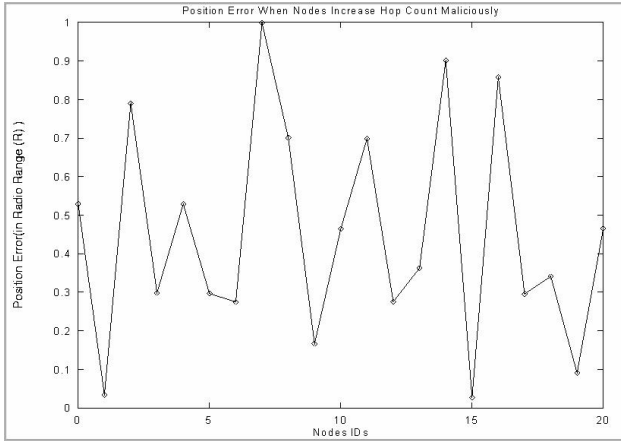


Figure 13: Position Errors When 10% Nodes Irregularly Increase Hop Count

The average position error for 21 nodes is $0.44R$. This error is relatively much smaller than that of compromised seed attack.

6.3. Hop Count Decrease Attack

To simulate this type of adversary, we randomly selected 10% of the nodes to decrease hop count by one instead of increasing hop count in the packet. The position error graph is shown in Figure 14.

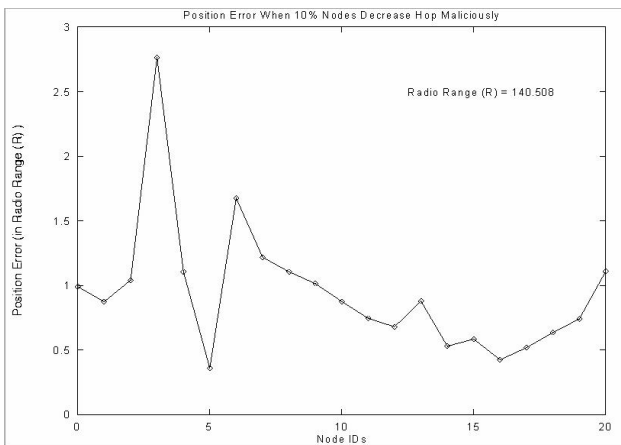


Figure 14: Position Errors When 10% Nodes Irregularly Decrease Hop Count

It is obvious from the results that decrease hop count attacks impact significantly on position accuracy. It produces an average position error $0.945R$, almost R in error rate. Henceforth, our security protocol which counteracts these attacks is very necessary.

6.4. Drop Packet Attack

Similarly for drop packet attacks, 10% nodes were chosen to drop a flooding packet each time it receives one. The results of position error under this type of attack are demonstrated in Figure 15.

This type of attack can cause an average position error around $0.69R$, which is less than decrease hop attack.

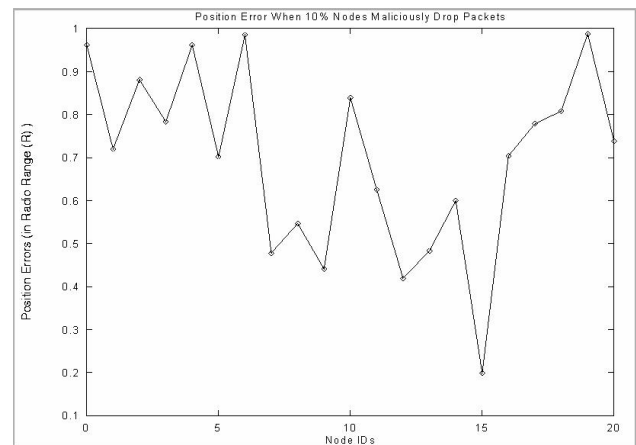


Figure 15: Position Errors When 10% Nodes Drop Packets on Purpose

Since our network model is small scaled with low density, therefore, the dropping packets effect is much more obvious. However, for large scale high density networks, the impact on position accuracy caused by dropping packets can be negligible.

6.5. Evaluation Summary

Demonstrated by simulation results, seed compromise attacks and attacks that compromised nodes decrease hop count can impact position accuracy most negatively in DV-Hop algorithm. The comparison between the multilateration and Grid Counter approach show that our proposed Grid Counter algorithm is much more immune to seed compromise. To prevent the compromised nodes' from dropping packets, the per-hop hashing scheme is used.

7. Conclusion

This paper has presented the security issues in the DV-Hop localization algorithm and the security protocols to countermeasure against the attacks. The protocols rely only on efficient symmetric cryptography. Rather than generally applying cryptography to an existing protocol, we carefully redesigned the protocol and its' processing. Our protocol can eliminate localization error caused by small portion of compromised nodes or seeds. Since many location dependent sensor network applications will operate in hostile environments, providing a way to increase confidence in the integrity of localization is a valuable design option.

References

- [1] Dragos Niculescu and Badri Nath. *DV Based Positioning in Ad hoc Networks*. Kluwer journal of Telecommunication Systems, 2003.
- [2] Radhika Nagpal, Howard Shrobe, and Jonathan Bachrach. *Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network*. In the 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03), Palo Alto, April, 2003.
- [3] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, Tarek Abdelzaher. *Range-free localization schemes for large scale sensor networks*. In the Proceedings of the 9th annual international conference on Mobile computing and networking (Mobicom 2003), 2003.
- [4] Andreas Savvides, Chih-Chieh Han, Mani B. Srivastava. *Dynamic fine-grained localization in Ad-Hoc networks of sensors*. In Proceedings of the 7th annual international conference on Mobile computing and networking.
- [5] Naveen Sastry, Umesh Shankar, David Wagner. *Secure verification of Location Claims*. ACM Workshop on Wireless Security, September 19, 2003.
- [6] B. H. Wellenhoff, H. Lichtenegger and J. Collins. *Global Positions System: Theory and Practice, Fourth Edition*. Springer Verlag, 1997.
- [7] Deborah Estrin and Ramesh Govindan and John S. Heidemann and Satish Kumar. *Next Century Challenges: Scalable Coordination in Sensor Networks*. Mobile Computing and Networking. 1999.
- [8] N. Bulusu, J. Heidemann and D. Estrin. *GPS-less Low Cost Outdoor Localization for Very Small Devices*. IEEE Personal Communications Magazine, October 2000.
- [9] P. Bahl and V. N. Padmanabhan. *RADAR: An In-Building RF-Based User Location and Tracking System*. In Proceedings of the IEEE INFOCOM '00, March 2000.
- [10] D. Niculescu and B. Nath. *Ad Hoc Positioning System (APS) using AoA*. In Proceedings of INFOCOM 2003.
- [11] Lingxuan Hu and David Evans. *Using Directional Antennas to Prevent Wormhole Attacks*. To appear at the 11th Annual Network and Distributed System Security Symposium (NDSS 2004).
- [12] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler and Doug Tygar. *SPINS: Security Protocols for Sensor Networks*. Wireless Networks Journal (WINE), September 2002.
- [13] Y. Hu, A. Perrig, and D. Johnson. *Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks*. Proceedings of INFOCOM 2003, IEEE, San Francisco, CA, April 2003.
- [14] Chris Karlof and David Wagner. *Secure Routing in Sensor Networks: Attacks and Countermeasures*. First IEEE International Workshop on Sensor Network Protocols and Applications, May, 2003.
- [15] X. Zeng, R. Bagrodia, and M. Gerla. *GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Networks*. Proceedings of the 12th Workshop on Parallel and Distributed Simulations, May 1998.