SECURE LOCALIZATION IN WIRELESS SENSOR NETWORKS

Yanchao Zhang, Wei Liu and Yuguang Fang
Department of Electrical and Computer Engineering
University of Florida, Gainesville, FL 32611
Email: {yczhang@, liuw@, fang@ece.}ufl.edu

ABSTRACT

Ad hoc wireless sensor networks (WSNs) have attracted a great deal of attention in recent years for their broad potential in both military and civilian operations. The proper operations of many WSNs rely on the knowledge of physical sensor locations. However, most existing localization algorithms developed for WSNs are vulnerable to attacks in hostile environments. As a result, adversaries can easily subvert the normal functionalities of location-dependent WSNs by exploiting the weakness of localization algorithms. In this paper, we first present a general secure localization scheme to protect localization from adversarial attacks. We then propose a mobility-assisted secure localization framework for WSNs.

I. INTRODUCTION

Many applications of wireless sensor networks (WSNs) require sensors to be aware of their physical locations, e.g., target detection and tracking, precision navigation and security surveillance. Driven by this demand, a number of localization algorithms have been proposed to provide per-node location information in WSNs, which can be classified into two categories: *range-free* such as [1]–[4] and *range-based* such as [5]–[8]. Both categories attempt to localize sensor nodes with the aid of a few *anchors* that are assumed to know their own locations, for instance, through GPS receivers or manual configuration. The former depends on individual sensors to estimate their locations based on the contents of messages received from neighboring sensors or anchors, while the latter relies on absolute point-to-point distance estimates or angle estimates between anchors and sensors. Although having less requirements on sensor hardware, range-free approaches only offer coarse-grained location accuracy [2], which makes them possibly suited for civilian applications but less suitable for military applications demanding high location accuracy. In this paper, we concentrate on range-based approaches for their capabilities in producing fine-grained locations.

This work was supported in part by the U.S. Office of Naval Research under Young Investigator Award N000140210464 and the US National Science Foundation under grant ANI-0093241 (CAREER Award).

We observe that almost all previous range-based proposals were designed for non-adversarial scenarios and thus are ill-suited for unattended and often hostile settings such as tactical military and homeland security operations. Under such circumstances, adversaries can easily subvert normal network functionalities by exploiting the weakness of localization algorithms to make sensors obtain wrong locations away from their true ones. For example, in security monitoring applications, sensor nodes, if improperly localized, cannot report the correct locations of intruders.

Motivated by this observation, our intention in this paper is not to provide any brand-new localization technique for WSNs, but to analyze and enhance the security of existing range-based approaches built upon Time-of-Arrival (TOA), the most commonly-used ranging technique. In particular, we make two main contributions. First, we propose a novel TOA-based secure localization scheme to protect localization from attacks. Second, we develop a mobility-assisted secure localization framework for WSNs.

The rest of this paper is structured as follows. Section II gives the adversary model under consideration and analyzes the security of the TOA technique. Section III presents our secure localization scheme called SLS. Section IV discusses our mobility-assisted secure localization framework for WSNs. Section V surveys the related work and this paper is finally concluded in Section VI.

II. PRELIMINARIES

A. Adversary Model

Adversaries in WSNs can be classified as either *external* and *internal* adversaries. The former have no access to the authentic keying material of the network and can be kept out of the network by effective authentication schemes. By contrast, internal adversaries might be either compromised nodes running malicious code or adversaries who stole authentic keying material from legitimate nodes. Internal adversaries can authenticate themselves to other legitimate nodes and thus are more difficult to defend against than external adversaries. We also assume that intrusion detection systems are imperfect, meaning that both types of adversaries are non-detectable at least for some time.

B. Security Analysis of TOA

In a WSN, there might be two types of nodes needed to be localized: common sensors and moving or static targets. For ease of presentation, we refer to a node to be localized as a ***locatee*** hereafter. We also focus our discussion on a general scenario that a set of anchors collectively determine the locations of certain locatees. A typical range-based localization algorithm with TOA works in two basic phases, (1) distance measurements between anchors and the locatee; and (2) data fusion that combines measurements to derive a location estimate. Adversaries usually launch attacks on phase one so as to affect the precision of locations calculated at phase two.

TOA [6] measures the signal arrival time, which can be directly translated into a distance estimate based on the known signal propagation speed. To eliminate the need for tight clock synchronization between anchors and locatees, a TOA two-way ranging method is often desired [9]. Assume that there are three anchors $A/B/C$. B first transmits at time t_1 a challenge to the locatee S and receives a response at time t_2. Then B can estimate its distance to S as $d_{sb} = (t_2 - t_1)c/2$, where c is the speed of light. Once anchors A and C obtain their distance estimates to S in the similar way, the well-known multilateration [6] technique can be applied to get a location estimate of S.

TOA is susceptible to attacks. For example, to make itself appear to be farther from anchor B than it actually is, an adversarial locatee can delay its response to B so as to enlarge t_2 and thus d_{sb}. Adversaries can achieve much the similar purpose by jamming and then replaying the response of an legitimate locatee. In addition, if used inappropriately, the challenge-response process would introduce opportunities for adversaries to carry out the *distance reduction attack*. For example, if adversarial, the locatee S might send a seemingly correct response to anchor B before receiving a complete challenge from B in order to reduce the TOA measurement at B and thus d_{sb}. Apparently, both distance enlargement and reduction attacks on TOA would lead to imprecise location estimates.

III. A Secure Localization Scheme

In this section, we present a general TOA-based secure localization scheme, called SLS. It allows a few anchors to securely localize a locatee, either adversarial or legitimate, in the presence of adversaries. For simplicity, this paper focuses on how to obtain a secure 2-D location estimate, but our scheme can be easily extended to the 3-D case.

A. Network and Trust Models

Let $\mathcal{A}$ denote the set of anchors and S be the locatee to be localized. Notice that the cardinality of $\mathcal{A}$, denoted by $n_a = |\mathcal{A}|$, should be at least 3 for determining a 2-D location. As is well known, the more anchors (distance estimates), the more precise location estimates are at the cost of increased overhead. Each anchor A_i ($1 \le i \le n_a$) is assumed to know its own location (x_i, y_i) at any time and place through GPS receivers or manual configuration or other means. For convenience, we assume A_1 to be the anchor leader in charge of the localization process, though each anchor should take turns to act as the leader in order to balance their resource usage in practice.

We also assume that all the anchors and S are within the transmission range of each other. We further assume the existence of an effective MAC protocol to support reliable radio transmissions among anchors and the locatee. Moreover, we assume that each pair of anchors share a pairwise secret key, based on which they can securely communicate with each other based on efficient symmetric-key algorithms. Hereafter, by saying a message is *securely transmitted*, we mean that the message is encrypted, integrity-protected and/or authenticated with the pairwise key shared between the transmitter and receiver. We further postulate that anchors are trusted and unassailable to adversaries, which is an assumption also made by previous work [10], [11]. Notice that this assumption is reasonable in that anchors are usually much fewer than locatees such as sensors. Therefore, we can spend more on them by enclosing them in high-quality tamper-proof enclosures and putting them under perfect monitoring. How to deal with compromised anchors is our ongoing work. In addition, we assume that the locatee S can establish a pairwise secret

Algorithm **SLS**: Generating a Valid Location Estimate

Inputs: $\mathcal{A}$: the anchor set, S: the locatee, K: a constant
Output: a location estimate (x_s, y_s) or (0, 0) if not found

1: A_1 broadcasts a "start" message
2: **for** each $A_i \in \mathcal{A}$ **do**
3: d_{si} = K-Distance(A_i, S, K)
4: send d_{si} securely to A_1
5: **end for** /*The following is done by A_1*/
6: $\mathcal{D} = \{(d_{si}, x_i, y_i), |1 \le i \le |\mathcal{A}|\}$, $\mathcal{C} = \{\mathcal{D}\}$
7: $m = |\mathcal{A}|$, $k = 1$, (x_s, y_s) = (0, 0)
7: **while** $\mathcal{C}! = \phi$ **do**
8: C_1 = GetOneWithoutReplacement($\mathcal{C}$)
9: (x_s, y_s) = CalPos(C_1)
10: **if** TestValid(C_1, x_s, y_s) == 1 **then**
11: break **while**
12: **end if**
13: (x_s, y_s) = (0, 0)
13: **if** (($--k$ ==0) && ($--m \ge 3$)) **then**
14: $\mathcal{C} \leftarrow$ all m-cardinality subsets of $\mathcal{D}$
15: $k = \binom{|\mathcal{A}|}{m}$
15: **end if**
16: **end while**
17: **return** (x_s, y_s)

Fig. 1. Algorithm **SLS**.

Algorithm **K-Distance**: Estimating the Distance to the Locatee

Inputs: A_i: an anchor, S: the locatee, K: number of rounds
Output: a distance estimate d_{si}

$\mathcal{T} = \phi$
for $(j = 1; j \leq K; j++)$ **do**
  A_i sends a random challenge nonce N_j to S
  S responds with N_j and another random nonce $N_{s,j}$
  A_i sets t_j = time elapses between challenge and response
  S sends to A_i a number $v = h(N_j \parallel N_{s,j} \parallel k_{s,i})$
  if $h(N_j|N_{s,j}|k_{s,i}) == v$ **then** /*by A_i*/
    $t_p = (t_j - t^{A_i}_{proc} - t^{S}_{proc} - t_{tran})/2$
    $\mathcal{T} = \mathcal{T} \cup \{t_p\}$
  end if
end for
t_{si} = median($\mathcal{T}$)
return $d_{si} = t_{si}c$ /*c is the light speed*/

Fig. 2. Algorithms **K-Distance**.

key $k_{s,i}$ with each anchor A_i. This can be accomplished by the method to be given in Section IV-B.

B. Overview

The operations of SLS are summarized in Fig. 1. The anchor leader A_1 triggers the protocol by broadcasting a "start" message. Then each anchor A_i including A_1 runs an algorithm called K-Distance to obtain a distance estimate d_{si} to the locatee S and securely transmits d_{si} to A_1. To avoid possible transmission collision, A_1 can schedule the activities of anchors through the "start" message. Once obtaining a set $\mathcal{D}$ ($|\mathcal{D}| = n_a$) of distance estimates from all peers, A_1 utilizes algorithm CalPos to derive a location estimate (x_s, y_s) and then tests its validity through algorithm TestValid. If the current (x_s, y_s) is valid, the algorithm finishes without further operations. Otherwise, A_1 re-executes CalPos with as input each of the (n_a-1)-cardinality subsets of $\mathcal{D}$ to get a new location estimate whose validity is also tested by using TestValid. If all the $(n_a - 1)$-cardinality subsets are traversed and still no valid distance estimate is generated, A_1 again runs algorithms CalPos and TestValid for each of the the $(n_a - 2)$-cardinality subsets of $\mathcal{D}$. This process repeats itself until either a valid distance estimate is found or all the 3-cardinality subsets of $\mathcal{D}$ are examined (3 is the minimum number of distance estimates required to derive a 2-D location estimate). If the latter case occurs without yielding a valid distance estimate, A_1 may consider that the localization process was subject to adversarial attacks and then take certain actions, e.g., reporting this abnormality to the control center, as stipulated by concrete network applications. In what follows, we will dwell on the aforementioned algorithms one by one.

C. K-Distance: a K-Round Distance Estimation Algorithm

Upon receiving a "start" message, each anchor A_i executes the algorithm K-Distance given in Fig. 2 to obtain a location estimate to the locatee S. A_i begins with sending to S a l-bit random nonce N_j and then starts a timer when the last bit of N_j is sent. When receiving such a challenge, S needs to immediately echo N_j concatenated by another l-bit random nonce $N_{s,j}$ generated by itself. To reduce the impact of processing delays, S should select $N_{s,j}$ from a set of pre-generated random nonces instead of generating one in real time. Subsequently, S sends to A_i a message authentication code $v = h(N_j|N_{s,j}|k_{s,i})$. Here h can be any computationally efficient hash function such as SHA-1 [12], $k_{s,i}$ is the pairwise secret key shared between S and A_i, and "|" represents the concatenation of messages.

When receiving the last bit of the response, A_i stops the timer and sets t_j equal to the elapsing time. Later, when v arrives, A_i can verify that the formerly received response indeed came from S by re-computing a keyed hash value and checking its equality to v. If so, A_i proceeds to calculate the one-way signal propagation time from itself to S as $t_p = (t_j - t^{A_i}_{proc} - t^{S}_{proc} - t_{tran})/2$ and appends t_p to an initially-empty set $\mathcal{T}$. Otherwise, t_j is simply dumped. $t^{A_i}_{proc}$ represents the time duration from when the last bit of the response hits the antenna of A_i until the response is completely decoded. Similarly, t^{S}_{proc} is the time duration from when the last bit of the challenge reaches the antenna of S until S transmits the first bit of the response. $t^{A_i}_{proc}$ and t^{S}_{proc} are device-dependent and usually are constant or vary in a tiny scale. Both can be pre-determined and preloaded to A_i to calibrate the time measurements to certain precision. Assume that transmission links from S to anchors have a bandwidth of b bps. Then the response transmission time t_{tran} is equal to $\frac{2l}{b}$ s.

K-Distance guarantees that adversaries cannot reduce t_p and thus the distance estimate $d_{si} = t_p c$ with non-negligible probability. One reason is that using message authentication code can ensure that an authentic response can only be sent by the locatee S. Another important reason is that nothing can travel faster than light so that S (if adversarial) would have great difficulties in sending the correct response before receiving the complete challenge. This is because the probability that S guesses a correct challenge N_j is $p_g = \frac{1}{2^l}$, which is often negligible when l is sufficiently large. For instance, if $l = 10$, $p_g \approx 1.5 \times 10^{-5}$.

Adversaries, however, can enlarge t_p and thus the distance estimate d_{si} by means of, for instance, delaying or jamming and replaying the response. If this happens, A_i would have no way to identify it. To mitigate such attacks, we require A_i to perform K times of distance estimations. The motivation is that adversaries might not be able to affect all K t_p values and thus distance estimates. Also notice that our method can help mitigate sporadic measurements errors. Here K is a design parameter that determines the tradeoff

between algorithm overhead and resilience to adversarial attacks and measurement errors. The next question is how to securely aggregate the resulting K time estimates in $\mathcal{T}$. The naive use of the average is insecure because adversaries can easily make the calculated average far away from the true one by enlarging just one time estimate to be sufficiently large.

As pointed out in [13], the median is a safer replacement for the average, so K-Distance uses the median of K time estimates to calculate d_{si}[1]. For brevity only, we assume $K \geq 3$ to be odd in what follows and the extension to the case that K is even is straightforward. Let $t_{(1)}$,..., $t_{(K)}$ denote trustful time estimates (without attacks) in $\mathcal{T}$ placed in an increasing order. We then have t_{si} = median($\mathcal{T}$) = $t_{(r)}$ for r equal to $\frac{K+1}{2}$. Consider first the simple case that adversaries enlarged just one time estimate from $t_{(j)}$ to $t'_{(j)}$. If $t_{(j)}, t'_{(j)} < t_{(r)}$, the median time estimate t_{si} remains unchanged; otherwise, it changes to some value between $[t_{(r-1)}, t_{(r+1)}]$. It is easy to see that K-Distance is vulnerable to single distance enlargement attack when K is equal to 1 (as all previous TOA-based proposals) or 2. In general, if m time estimates were enlarged, t_{si} either remains unchanged or changes to some value between $[t_{(r-m)}, t_{(r+m)}]$, depending on how adversaries contaminated the time estimates. It is obvious that the median method can tolerate the enlargement of up to about half of the time estimates. This may be enough for defending against less powerful adversaries in most cases. However, if K assumes a small value and adversaries launch persistent attacks, the final median time estimate t_{si} might be enlarged to be an arbitrarily large value when m is greater than $\frac{K+1}{2}$. Fortunately, we can deal with this worse situation using the TestValid algorithm that will be explained shortly.

D. CalPos: Calculating a Location Estimate

After collecting distance estimates from all peers, A_1 is able to calculate a location estimate (x_s, y_s). Remind that each anchor A_i has a 2-D location (x_i, y_i) known to A_1. For each distance estimate d_{si}, we then have

$$f_i(x_s, y_s) = d_{si} - \sqrt{(x_s - x_i)^2 + (y_s - y_i)^2}.$$

A ML location estimate can be obtained by taking the MMSE of a system of $f_i(x_s, y_s)$ equations, i.e., minimizing $F(x_s, y_s) = \sum_{i=1}^{n_a} f_i^2(x_s, y_s)$. There are many methods to compute MMSE estimates. A popular one is the Taylor-Series method given in [14].

[1]We notice that there might exist other methods such as Least Median Squares (LMS) to deal with outliers (distance estimates enlarged in our case). However, they are less computationally efficient than the median method.

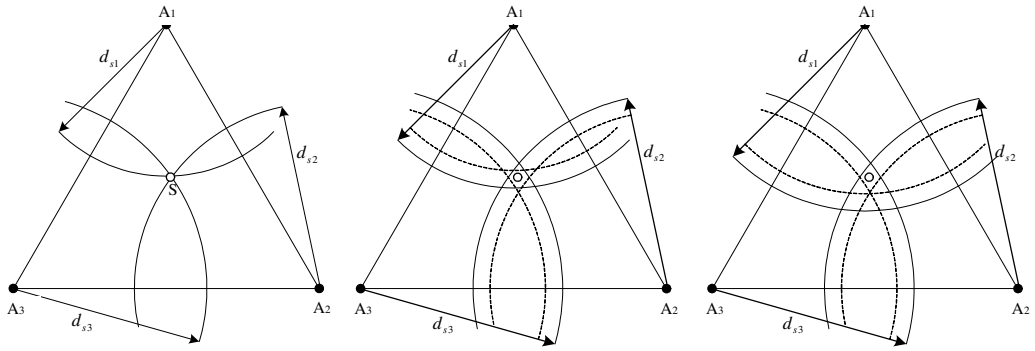
E. TestValid: Testing the Validity of Location Estimates

TestVaild is executed by A_1 to test the validity of a location estimate (x_s, y_s). This protocol is necessary because K-Distance cannot completely prevent distance enlargement attacks and thus some distance estimates used by CalPos to generate (x_s, y_s) might have been maliciously enlarged by adversaries.

Consider first the simple case that there are no measurement errors. If all the n_a distance estimates were not enlarged by adversaries, (x_s, y_s) should be exactly the intersection point of n_a circles $\{(x - x_i)^2 + (y - y_i)^2 = d_{si}^2 | 1 \leq i \leq n_a\}$. See Fig. 3(a) for an example with three anchors. In the presence of adversaries, we just need to check whether the location estimate is inside the n_a-vertex polygon formed by all anchors. The underlying logic is very simple. If adversaries want to make S appear to be at any location other than its true location, they have to enlarge certain distance estimates, while at the same time reduce certain distance estimates to keep the bogus location inside the triangle. As mentioned before, however, our K-Distance algorithm can prevent adversaries from launching distance reduction attacks. Therefore, anchors can be assured that the location estimate is trustable as long as it resides in the n_a-vertex polygon.

To determine the inclusion of a point inside a polygon, we select the well-known *ray-tracing* method [15], denoted by IfInside, for its simpleness and computational efficiency. This method works by starting at the point in question and drawing a straight line in any direction. If the number of times the ray intersects the polygon edges is odd, the starting point is inside the polygon and is outside otherwise. It is easy to understand intuitively. Each time the ray crosses a polygon edge, its in-out parity changes since each edge always separates the inside of a polygon from its outside. Eventually, any ray must end up beyond and outside the bounded polygon. Therefore, if the point is inside, the sequence of crossings "→" must be: in→out→ $\cdots$→in→out, and there are an odd number of them. Similarly, if the point is outside, there are an even number of crossings in the sequence: out→ $\cdots$→in→out.

In practical scenarios, however, time measurement errors and thus distance estimate errors occur inevitably. As a result, the n_a circles centered at anchors will not have a common intersection point, but form an intersection area in which the location estimate is located. This would introduce opportunities for adversaries to launch attacks on localization. Consider again the example with three anchors shown in Fig. 3(b). Suppose the distance estimate d_{s1} was maliciously enlarged, while the other two d_{s2} and d_{s3} were just enlarged because of measurement errors. It is obvious that, by adjusting the level of enlarging d_{s1}, adversaries might



(a) No measurement errors. (b) Measurement errors exist. (c) d_{s1} is maliciously enlarged.

Fig. 3. An example of location validity test with three anchors.

Algorithm **TestValid**: Testing the Validity of a Location Estimate

Inputs: $\mathcal{B}$: an anchor set, (x_s, y_s): a location estimate
Output: 1 if valid, else 0

```
1: v = 0
2: if IfInside(B, x_s, y_s) then
3:     v = 1
4:     for (i = 1; i ≤ |B|; i++) do
5:         if (d_si − sqrt((x_s − x_i)^2 + (y_s − y_i)^2) > δ) then
6:             v = 0
7:             break for
8:         end if
9:     end for
10: end if
11: return v
```

Fig. 4. Algorithm **TestValid**.

be able to freely enlarge the intersection area of the three circles and thus make the ML distance estimate (though still inside the triangle) deviate much from the true location of the locatee S. Fortunately, we can efficiently mitigate this attack by imposing certain reasonable constraints on the ML distance estimate. Define δ to be the two-sided maximum allowable measurement error with regard to distance estimates. Now (x_s, y_s) should be within the intersection area of n_a rings $\{(d_{si} - \delta)^2 \leq (x - x_i)^2 + (y - y_i)^2 \leq d_{si}^2 | 1 \leq i \leq n_a\}$. It means that, after calculating a ML distance estimate through CalPos, we need to further check if[2] $d_{si} - \sqrt{(x_s - x_i)^2 + (y_s - y_i)^2} \leq \delta$ holds for each distance estimate d_{si}. If so, (x_s, y_s) is considered valid and invalid otherwise. With our method in place, adversaries might only be able to enlarge d_{si} a little bit to make the resulting (x_s, y_s) appear to be valid, leading to a tolerable location imprecision. Instead, if they enlarge d_{si} by a relatively large amount, the resulting (x_s, y_s) would be detected as invalid. Therefore, although our method cannot completely eliminate distance enlargement attacks, which is believed to be impossible for any security mechanism, it does very much constrain the impact of adversaries to an acceptable level. It is worth pointing out that the above method only makes sense when the location estimate is inside the n_a-vertex polygon. This is easily understandable by taking a close look at Fig. 3(c). Since the two rings centered at A_2 and A_3 respectively have two intersection areas, one inside the triangle and the other outside, adversaries might be able to enlarge d_{s1} by a large amount so as to make the three rings intersect with each other outside the triangle and thus result in a seemingly valid location estimate. Therefore, IfInside needs to be executed at any time. The complete location validity test process is summarized in Fig. 4.

[2]Notice that the left part of the inequality is always no less than zero because the ML method always yields a location estimate inside the circle with radius d_{si}.

IV. Secure Localization in WSNs

In this section, we demonstrate the use of SLS in heterogenous WSNs, where a few anchors have known locations and are much more powerful than common sensors in terms of computational capacities and energy resources.

A. Overview

We assume that there are totally $N_a = n_g \cdot n_a$ mobile anchors divided into $n_g (\geq 1)$ groups of size $n_a (\geq 3)$. Here n_g is a design parameter that determines the trade-off between network bootstrapping delay and localization overhead: the larger n_g, the smaller bootstrapping delay, the more mobile anchors are needed, and vice versa. Examples of mobile anchors include mobile robots, Unmanned Aerial Vehicles (UAVs) flying at low level, or even persons carrying wireless devices. Also notice that mobile anchors can undertake other important tasks besides sensor localization, e.g., acting as data mules to collect sensor data or message ferries to improve data delivery performance in large-scale sensor networks. We also assume that anchors and sensors have the same transmission range r_0.

Our scheme consists of three steps: (1) each anchor of an n_g-member group obtains a distance estimate to a sensor

to be localized through the aforementioned two-way TOA ranging method; (2) anchors collaboratively run SLS to get a valid location estimate; and (3) if found, the location estimate is securely transmitted to the target sensor. Notice that traditional sensor localization methods such as AHLos [6] require each sensor to measure distances to anchors and then perform multilateration to get a location estimate by itself. In contrast, our scheme shifts the resource-hungry ranging and computation tasks to a few powerful anchors so that individual sensors no longer need to possess precise ranging and powerful computation capabilities and thus can be made much cheaper. It also justifies the previous assumption on the tamper-proofness of anchors as required by SLS. The reason is that, anchors are rarer as compared to common sensors and hence we can spend more on them, so it may be feasible to enclose them in high-quality tamper-proof enclosures and/or put them under perfect monitoring. More important, our scheme enables the secure localization in the face of adversaries, which is a mission impossible to be undertaken by resource-constrained sensors themselves.

B. Mobility-Assisted Sensor Localization

Each anchor group is instructed to perform strategic group pause/movement across the sensor field, during which group members always maintain a physically n_a-vertex regular polygon[3] with the longest distance between any two vertices equal to r_0. For example, if $n_a = 4$ and $n_g = 1$, then mobile anchors form a square with side length $\frac{\sqrt{2}r_0}{2}$. Starting from the left bottom of the sensor field, the anchor group can move $\frac{\sqrt{2}r_0}{2}$ upward each time, pause for a while until sensors in the square are all securely localized through SLS, and then move upward again until reaching the upper boundary of the sensor field[4]. Then the anchor group makes a horizontal right shift for $\frac{\sqrt{2}r_0}{2}$ and starts moving downward with one-time moving distance $\frac{\sqrt{2}r_0}{2}$. This process continues until all N sensors in the field are securely localized. It is worth pointing out that the one-time moving or shifting distance in reality should be less than $\frac{\sqrt{2}r_0}{2}$ to guarantee that sensors near polygon edges can be correctly localized. The remaining problem is how to securely transmit calculated location estimates from the anchor group leader to individual sensors. For this purpose, we need an efficient method to establish pairwise shared secret keys between anchors and common sensors.

Assume that there is a master key κ_{ma} known only to the network planner and tamper-proof anchors. Before deployment, each sensor i is preloaded with an ID-based individual key $IK_i = h(i \parallel \kappa_{ma})$ corresponding to its unique ID, where h can be any computationally efficient hash functions such as SHA-1 [12]. During the network operation, when one anchor, say A_1, wants to securely communicate with sensor i, it can generate IK_i on the fly to be used as the pairwise key shared with node i. This pairwise key establishment method is very efficient in that each sensor only needs to memorize its own ID-based key and the computational overhead is often negligible since anchors just need to execute highly efficient hash functions. Below is described the whole localization process.

Whenever pausing after one movement, the anchor leader, say A_1, announces the group existence by broadcasting a "helloLocation" message. Due to the shared wireless medium, all the sensors inside the transmission range of A_1 will hear the "helloLocation" message, of which some might already have been properly localized. Only those who have not been localized yet will respond to the message. Among them, most sensors are inside the anchor polygon, while a few others might be outside. The latter would be detected by A_1 when running the IfInside test. It is worth noting that, if all the neighboring sensors of A_1 simultaneously send replies to A_1, possible MAC-layer collisions may occur. For simplicity, we assume the reliable transmission of such replies in this paper. It can be achieved for instance through MAC-layer retransmissions or by using a random jitter delay for which each sensor has to wait before responding to a "helloLocation" message. Once collecting all responses, A_1 together with its group peers proceeds to localize corresponding sensors one by one. Below is shown the complete process for securely localizing a sensor i inside the anchor polygon:

$$
\begin{array}{lcll}
A_1 & \xrightarrow{\text{broadcast}} & \text{node } i & : \text{“helloLocation”} \\
\text{node } i & \xrightarrow{\text{unicast}} & A_1 & : i, time, h(i \parallel time \parallel IK_i) \\
A_1 & \xrightarrow{\text{unicast}} & \text{node } i & : \{i, (x_i, y_i)\}_{IK_i}
\end{array}
$$

Sensor i responds to the "helloLocation" message by unicasting to A_1 a message including its ID i, a timestamp and a keyed hash value of the former two with any efficient hash function h such as SHA-1 [12]. After receiving i's response, A_1 first generates IK_i and then verify that the response did come from sensor i by calculating a keyed hash value and checking its equality to the received one. This step is indispensable because otherwise adversaries might deceive A_1 and hence the anchor group into wasteful localization operations. Once authenticating sensor i, A_1 together with its group peers runs the SLS algorithm to derive a secure location estimate (x_i, y_i). After that, A_1 unicasts $\{i, (x_i, y_i)\}_{IK_i}$ back to node i, where $\{\cdot\}_k$ denotes an encryption operation with key k using any efficient symmetric encryption algorithm such as RC6 [16].

[3]Other polygon shapes might be used as well as long as a full coverage is guaranteed.

[4]Some anchors may need to move outside the sensor field a little bit to ensure that boundary nodes are properly localized.

Subsequently, node i can decrypt the ciphertext with the preloaded IK_i. Also notice that, in the third step above, A_1 might pack together the responses (if any) for several sensors and broadcast them in one message to target sensors so as to reduce the communication overhead.

The purpose of embedding i in the ciphertext is to withstand the attack that an adversary may send a forged location to node i. Since adversaries do not have the knowledge of IK_i, they cannot form the appropriate ciphertext which would be decrypted to produce the correct field i. As a result, if its own ID i does not match the first field of the decrypted result, node i should discard the packet because it might come from an adversary. Otherwise, i accepts the packet and thus the location inside.

V. Related Work

In this section, we briefly review some important work that is closely related to this paper. Brands and Chaum [17] proposed a TOA-based distance bounding protocol that can be used to verify the proximity of two devices connected by a wired link. Sastry *et al.* [18] proposed a similar distance bounding approach based on ultrasound and RF signals to verify the presence in a region of interest instead of the exact location of a wireless device. The same problem was also addressed later by Vora and Nesterenko [19]. More recently, Lazos and Poovendran [10] proposed an approach to secure range-free localization techniques [1]–[4] for sensor networks. By contrast, this paper concentrates on securing range-based localization techniques [5]–[8]. The closest work to ours SLS can be found in [11], in which a scheme called Verifiable Multilateration (VM) was proposed for secure positioning of wireless devices. Similar to our proposed SLS, VM also works by first obtaining distance estimates to the device to be localized and then deriving a MMSE location estimate. However, SLS differs significantly from VM in two major aspects. First, the K-Distance algorithm used by SLS is able to mitigate the impact of adversarial attacks and sporadic measurement errors in the first place, which is a nice property not provided by VM. In fact, the distance bounding process in VM can be treated as a special case of K-Distance when $K = 1$, despite the different message formats. Second, VM calculates location estimates on the basis of three anchors or triangles. By contrast, we consider a more general case by using n_a-vertex polygon formed by n_a anchors for $n_a \geq 3$.

VI. Conclusion

In this paper, we analyze the security of TOA-based localization techniques proposed for WSNs. We also present a novel secure localization algorithm, called SLS, to enable secure localization in the presence of adversaries. In addition, we develop a mobility-assisted scheme to apply SLS in WSNs. As the future research, we plan to extend our approach to range-free localization techniques. We also intend to further investigate the potentials of location information in securing WSNs, see for example [20].

References

[1] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low cost outdoor localization for very small devices," *IEEE Personal Commun. Mag.*, vol. 7, no. 5, pp. 28–34, Oct. 2000.

[2] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher, "Range-free localization scheme in large scale sensor networks," in *ACM MOBICOM'03*, San Diego, CA, Sep. 2003.

[3] D. Niculescu and B. Nath, "DV based positioning in ad hoc networks," *Journal of Telecommunication Systems*, 2003.

[4] L. Hu and D. Evans, "Localization for mobile sensor networks," in *ACM MOBICOM'04*, Philadephia, PA, Sep/Oct 2004.

[5] L. Doherty, K. S. Pister, and L. E. Ghaoui, "Convex optimization methods for sensor node estimation," in *IEEE INFOCOM'01*, Anchorage, Alaska, April 2001.

[6] A. Savvides, C. Han, and M. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *ACM MOBICOM'01*, Rome, Italy, July 2001.

[7] D. Niculescu and B. Nath, "Ad hoc positioning system (APS) using AoA," in *IEEE INFOCOM'03*, San Francisco, CA, April 2003.

[8] X. Cheng, A. Thaeler, G. Xue, and D. Chen, "TPS: A time-based positioning scheme for outdoor wireless sensor networks," in *IEEE INFOCOM'05*, Miami, FL, March 2005.

[9] D. D. McCrady, L. Doyle, H. Forstrom, T. Dempsey, and M. Martorana, "Mobile ranging using low-accuracy clocks," *IEEE Trans. Microwave Theory Tech.*, vol. 48, no. 6, pp. 951–957, June 2000.

[10] L. Lazos and R. Poovendran, "Serloc: Secure range-independent localization for wireless sensor networks," in *ACM WiSe'04*, Philadelphia, PA, Oct. 2004.

[11] S. Čapkun and J.-P. Hubaux, "Secure positioning of wireless devices with application to sensor networks," in *IEEE INFOCOM'05*, Miami, FL, March 2005.

[12] NIST, "Digital hash standard," Federal Information Processing Standards PUBlication 180-1, April 1995.

[13] D. Wagner, "Resilient aggregation in sensor networks," in *ACM SASN'04*, Washington, DC, Oct. 2004.

[14] W. Foy, "Position location solutions by taylor-series estimation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 12, no. 2, pp. 187–194, March 1976.

[15] W. R. Franklin, "Pnpoly - point inclusion in polygon test." [Online]. Available: http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html

[16] R. Rivest, M. Robshaw, R. Sidney, and L. Yin, "The rc6 block cipher," v1.1, Aug. 1998. [Online]. Available: http://www.rsasecurity.com/rsalabs/rc6/.

[17] S. Brands and D. Chaum, "Distance-bounding protocols (extended abstract)," in *Theory and Application of Cryptographic Techniques*, 1993, pp. 344–359.

[18] N. Sastry, U. Shankar, and D. Wagner, "Secure verification of location claims," in *ACM WiSe'03*, San Diego, CA, Sep. 2003.

[19] A. Vora and M. Nesterenko, "Secure location verification using radio broadcast," in *OPODIS'04*, Grenoble, France, Oct. 2004.

[20] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Securing sensor networks with location-based keys," in *IEEE WCNC'05*, 2005.