Detection of the Evil Ring Attack in Wireless Sensor Networks Using Cross Verification

Wei Shi
Faculty of Business and Information Technology
University of Ontario Institute Technology
Oshawa, Canada
Email: wei.shi@uoit.ca

Michel Barbeau
School of Computer Science
Carleton University
Ottawa, Canada
Email: barbeau@scs.carleton.ca

Jean-Pierre Corriveau
School of Computer Science
Carleton University
Ottawa, Canada
Email: jeanpier@scs.carleton.ca

***Abstract*—In ad hoc networks and wireless sensor networks, several routing algorithms rely on the knowledge by the network nodes of their own geographic location and those of others. For cases where a node doesn't have its own positioning device (e.g., GPS), Alfaro et al. propose several algorithms that a node can run to determine its geographic position using position reports from neighbors.**

In this paper, we first present the *evil ring attack*, an attack on the geographic location algorithms of Alfaro et al. that misleads nodes about the true position of their neighbors. An attacker sends false reports with a position that sits on a circle centered at the victim's location and of a radius equal to the distance between the victim and attacker. The attack succeeds because the calculation of the distance between the victim and attacker is not affected despite this fake position. We then present and analyze an evil ring attack detection algorithm in which a position-unaware sensor node crosschecks the consistency of the information it collects from its neighbors with the information collected by other trusted neighbors. This algorithm detects the existence of neighbors running the *evil ring attack*.

We propose a general distributed algorithm for a) localizing sensors in a wireless sensor network in the presence of some malfunctioning ones, and b) detecting such malfunctioning sensors.

***Keywords*-Wireless Sensor Network; Liar Detection; Localization; Algorithms.**

I. Introduction

In ad hoc networks and wireless sensor networks (WSNs), several routing algorithms rely on the knowledge, by network nodes, of their own geographic location and geographic locations of others. Such routing algorithms include compass routing, face routing [1] and geographical routing [2]. Nodes equipped with GPS devices can determine their geographic location. There are, however, several instances where a GPS device may be unavailable or inoperative because, for instance, of signal obstruction. Alfaro et al. proposed several algorithms that a node can run to determine its geographic position using position reports from neighbors and geographic location techniques such as time of arrival, time difference of arrival and angle of arrival [3], [4]. In such work, it is assumed that some nodes may be malfunctioning or malicious and, consequently, may not report correct positions. These nodes are called *liar*s. The algorithms are designed such that every node not equipped with a GPS device can determine its location using position reports from neighbors even in the presence of *liar*s. This is possible via the application of majority rules, as long as the number of *liar*s is below a certain threshold.

In [5], a mechanism for secure computation and verification of positions of wireless devices was presented. This method is robust and resists against distance modification attacks from a large number of attacker nodes, but is not able to detect and filter out the attackers. In [6] the authors proposed a secure localization mechanism that detects the existence of attacker nodes, termed phantom nodes, without relying on any trusted entity. The approach is limited to stochastic guarantees, its main drawback. A decentralized method that solves both secure localization (i.e., determining the location of nodes in the presence of malicious adversaries) and location verification (i.e., verifying the location claimed by a node) was studied in [7]. It requires a small number of reference points (locators) and it limits the ability of an adversary to spoof a sensor's location. This method, however, cannot detect attackers in WSNs.

In this paper we present an attack on the geographic location algorithms of Alfaro et al. that misleads nodes about the true position of their neighbors. A *liar* can send a false position. This attack will not be detected as such as long as the *liar* sits on a circle centered at the victim's location and of a radius corresponding to the distance between the *liar* and the victim. Such an attack succeeds because the calculation of the distance between the *liar* and victim is not affected. Thus, victims can still calculate their position correctly, but are unable to detect the false position reports received from *liar*s. In turn, this attack enables attacks against routing protocols requiring knowledge of positions of neighbors and other nodes in the networks to operate correctly. We call this stratagem the *evil ring attack*.

Building on the work of Alfaro et al. [3], [4] and Delaet

978-1-4244-7265-9/10/$26.00

et al. [8], we propose here a distributed algorithm for localizing sensors in WSNs in the presence of *liar*s and for detecting *evil ring attack*s. More specifically, in a paper on the problem entitled *Deterministic secure positioning*, Delaet et al. present an algorithm that requires a priori knowledge of the position of each sensor in the network. That algorithm requires $2n^2$ messages but clearly imposes a demanding, if not problematic, constraint in the form of a priori knowledge. It is our goal to avoid this hurdle in our proposed solution. We present an algorithm in which a location-unaware sensor crosschecks the consistency of the information it collects from its neighbors with the information collected by other trusted neighbors. The existence of neighbors running the *evil ring attack* can be detected with our proposed algorithm. Figure 1 shows a group of ten sensors located on a Google map. Assume sensors 1 and/or 2 and/or 3 are *liar*s. Our algorithm allows location-unaware sensors A and B to calculate their location according to the locations that their neighboring sensors report and to identify the *liar*s among their neighbors.

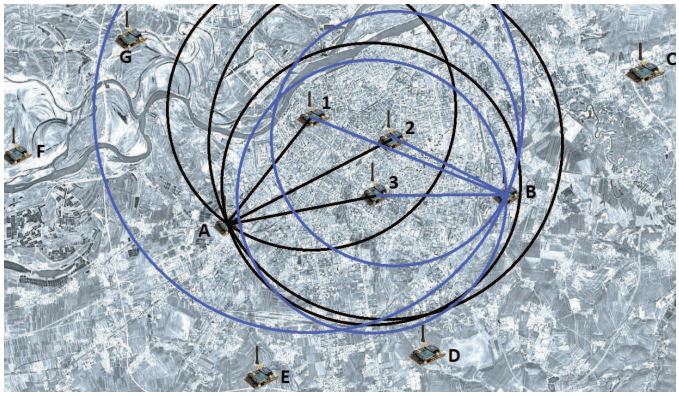

Figure 1. A group of ten sensors displayed on a Google map. Location-unaware sensors A and B calculate their location despite the fact there are *liar*s among their neighbors. They crosscheck the consistency of the information they collect from neighbors in order to identify the *liar*s using Algorithm *Cross Check*.

The details of the *evil ring attack*, in the context of the work of Alfaro et al., are presented in Section II. Our model and assumptions are formally presented in Section III. The *evil ring attack* detection algorithm is described in Section IV and analyzed in Section V. We conclude with Section VI.

II. Attack Model: Evil Ring

Alfaro et al. described algorithms that a node U_i can apply to determine its own position by obtaining the positions of its one-hop neighbors [3], [4]. More precisely, in these algorithms, node U_i determines its location from the positions of neighbors, considered three at a time. Let $V1$, $V2$ and $V3$ denote three such neighbors of U_i with their calculated distances d_1, d_2 and d_3 respectively. With respect to these three neighbors, the position of U_i is determined by the point at the intersection of the three circles centered at positions $V1$, $V2$ and $V3$ and of radii d_1, d_2 and d_3 respectively.

Algorithm 1 in Ref. [4], referred to as algorithm *Majority-ThreeNeighborSignals*, uses a majority rule. Let us elaborate. Given the positions of all one-hop neighbors of node U_i (as reported by these neighbors of U_i), triples of positions are created for all combinations of these neighbors. An intersection point is calculated for each such triple. The number of occurrences of each intersection point resulting from all triples is then counted. The algorithm succeeds in localizing U_i if there is a consensus, that is, if more than half of the total number of triples compute the same intersection point (which becomes the resulting position of U_i).

The interest of such algorithms resides in the fact that it is assumed that some neighbors may lie about their true position, but not about their distance, and still the proposed algorithms work. Thus, such algorithms are said to be *liar* tolerant. Most importantly, Alfaro et al. provide an upper bound on the number of *liar*s that the algorithm can tolerate whilst working correctly.

Let us now describe the *evil ring attack* in this context, that is, a situation in which *liar*s provide false positions that cannot be detected by the majority-rule used in this algorithm. (This attack is an enabler for other attacks, such as attacks against position-based routing protocols.) The attack is pictured in Figure 2. Let us assume that node $V1$ is a *liar* (as opposed to a truth teller) and is used to determine the position of U_i. Part *a* of Figure 2 shows that node $V1$ can report any position located on a (dashed) circle centered at the position of U_i and of radius d_1. The calculation of the distance to node $V1$ by U_i is not affected and is consistent with distances calculated using position reports from truth tellers. However, assume node U_i is misled by $V1$, which reports a wrong position to U_i. This false position disrupts the expected operation of any position-based algorithm, such as those used for geographical routing. Parts *b* and *c* of Figure 2 show that the attack can involve two or three independent *liar*s in a single triple.

We describe, in the following section, a technique with which such *liar*s can be detected by means of a cross verification of position reports with trusted neighbors.

III. Model and Assumptions

Let $\mathcal{V}$ denote the collection of sensors nodes in some area, with $|V| = n$. Let $\mathcal{M}$ denote a group of malfunctioning sensor nodes, with $|\mathcal{M}| = m$ and $\mathcal{M} \subset \mathcal{V}$. Each such malfunctioning sensor is called a *liar* because it does not report its position (i.e., coordinates) correctly. In this case, a *liar* is said to report a fake position. The intent of a *liar* may be *malicious*, in that case the *liar* may mislead the node it reports its position to into a wrong location calculation. Alternatively, the intent of a *liar* may be *unintentional* in the sense that obstacles or other physical circumstances (e.g.,

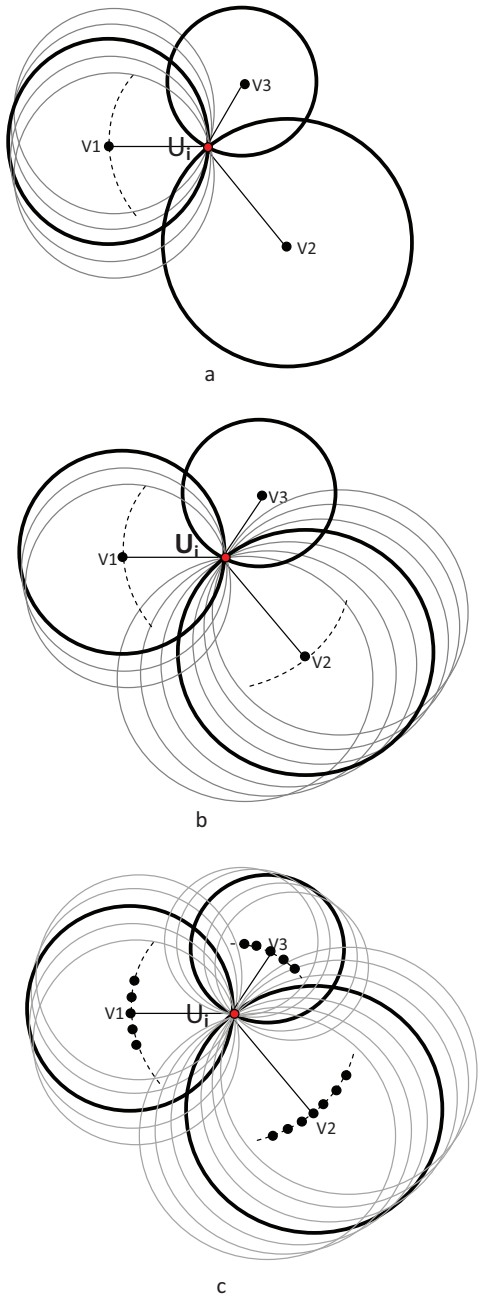

Figure 2. *Evil ring attack* involving one (part *a*) or two (part *b*) or three (part *c*) sensor nodes in a single triple.

multi-path interference) prevent a sensor from reporting its correct location [3]. We assume that the *liar*s cannot corrupt the measurement techniques used to determine the distance d between two sensor nodes. We define $U_i \in \mathcal{U}$ ($|\mathcal{U}| = k$) as a sensor node that does not know its location. We refer to U_i as a *location unaware* sensor node. We also assume that no three sensors are colinear, and that there exists no sensor U_j positioned on a line that is orthogonal to the line passing through a node $v \in \mathcal{M}$ and v's fake position. We prove that the execution our algorithm makes U_i become cognizant of both its location and of nodes carrying out the *evil ring attack*.

Before we describe the details of our algorithm, let's prove the following theorem:

Theorem 1: Algorithm *Majority-ThreeNeighborSignals* presented in [3] is not sufficient to isolate *liar*s that carry out the *evil ring attack*.

Proof:

As stated in [3], [8], a $U_i \in \mathcal{U}$ should be able to calculate a position, if it is given locations of any three truthful sensors (i.e., sensors that are not *liar*s and whose positions are known a priori), and the distances to U_i. As illustrated using circles whose circumferences are drawn solid (as oposed to dashed) in Figure 3, if $V1$, $V2$ and $V3$ are not *liar*s and the distances from $V1$, $V2$, $V3$ to U_i are measured respectively as $d(V1, U_i)$, $d(V2, U_i)$ and $d(V3, U_i)$, U_i can calculate its location (X_i, Y_i) by resolving the following three equations:

1) $(V1x - X_i)^2 + (V1y - Y_i)^2 = d(V1, U_i)^2$

2) $(V2x - X_i)^2 + (V2y - Y_i)^2 = d(V2, U_i)^2$

3) $(V3x - X_i)^2 + (V3y - Y_i)^2 = d(V3, U_i)^2$

If at least one sensor in the triple $V1$, $V2$ and $V3$ is a *liar*, then U_i either calculates a wrong position (see Figure 3) or fails to calculate a location. Figures 3 and 4 show how the calculation by U_i of the circle associated with node $V3$ is affected when the latter provides a wrong position $V3'$. Node U_i detects *liar*s following two simple rules:

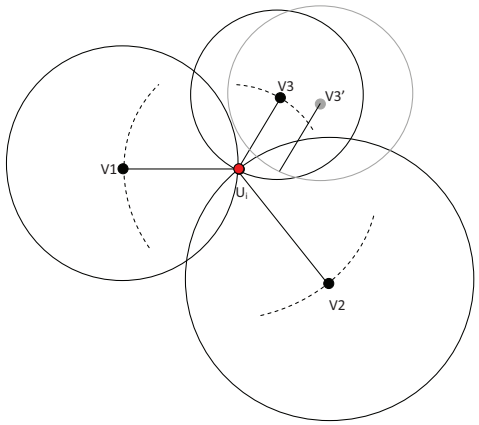

Figure 3. Node $V3$ reporting a fake position $V3'$.

1) Rule 1: If U_i calculates a position (X_i', Y_i') (according to some triple of sensor nodes) and this position does not match the real position (X_i, Y_i) of U_i, or if U_i fails to calculate its position, then U_i adds the sensors in the triple to the list of *liar*s that it keeps.
2) Rule 2: Conversely, if U_i calculates a position (X_i', Y_i') (according to some triple of sensor nodes) and this

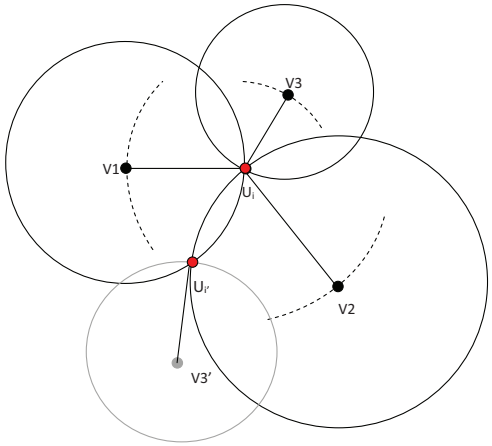

Figure 4. Node $V3$ reporting a fake position $V3'$: Node U_i infers a wrong position U_i'.

position does match the real position (X_i, Y_i) of U_i, then U_i adds the sensors in the triple to the list of truth tellers that it keeps.

Now let us consider the following situation (see Figure 2): As in Figures 3 and 4, $V1$, $V2$ and $V3$ are three neighbor sensor nodes of U_i, and each of them knows its location. Assume $V1$ is a *liar* and that its fake position (i.e., a pair of coordinates that it uses to fool U_i) is on a circle $\mathcal{C}$ represented by a dashed line, which centers on U_i and has $d(V1, U_i)$ as its radius. As shown in Figure 2 - a, all the gray circles intersect at U_i. Thus, U_i is able to calculate a position (X_i', Y_i'), identical to (X_i, Y_i), as though $V1$ never lied about its position. According to Rule 2 mentioned earlier, U_i should conclude that this group of sensor nodes are not *liar*s. This contradicts our assumption. Hence, in this case, Algorithm *Majority-ThreeNeighborSignals* is not sufficient to detect the *liar*s. It is trivial to prove the equivalent cases in which $V2$ or $V3$ is a *liar* instead of $V1$. Similarly, we can generalize the proof to when two out of these three sensor nodes are *liar*s (see Figure 2 - b) or when the three of them are all *liar*s (see Figure 2 - c). Thus, we can conclude that, indeed, Algorithm *Majority-ThreeNeighborSignals* cannot be used to detect *liar*s that perpetrate the attack model that we call evil ring. ■

In the following section, we describe an algorithm that can detect the *evil ring attack*. A sensor node U_i uses another trusted sensor node $U_j \in \mathcal{U}$ to cross check its information. We then prove the correctness of this algorithm and analyze its communication cost.

IV. Evil Ring Attack Detection Algorithm

There are two major steps in our *evil ring attack* detection algorithm: a) Location Request and b) Cross Checking:

- Step 1 - Location Request: In the location request step, k sensors that do not know their coordinates (i.e., location-unaware sensors) initially send requests to all the other sensors in the area. Then the sensors (both *liar*s and truth tellers) that are aware of their location send back their coordinates. Each of these k location-unaware sensors calculates it coordinates (using every possible combination of three of its reporting neighbor sensors) and the distances between itself and each of these three sensors. Each of these k location-unaware sensors decides its own position based on majority voting as described in [3]. A group of three sensors, denoted as triple t, is immediately listed as *liar*s if, from it, i) a location-unaware sensor cannot calculate a position or ii) the position obtained from this triple does not match the real position of this sensor (as established through majority voting). Otherwise this location-unaware sensor keeps this triple t in a *Cross-Check* list. This list is verified in the next step.
- Step 2 - Cross Checking: The k location-unaware sensors broadcast their *CrossCheck* list (that consists of triples). Each triple of the *CrossCheck* list of a location-unaware sensor s has its sensors identified as *Truth Teller*s if they all belong to and are consistent with at least one other trusted *CrossCheck* list that s receives. Otherwise this triple is put into the *liar* list of s.

The following is the pseudocode for Algorithm *Cross Check*.

Algorithm 1 Cross Check

1: Node U_i requests the location of its neighbors.
2: $\forall\ v \in \mathcal{V}$ sends its location to U_i
3: For each triple t of neighbors V_i, V_j, V_k in $\mathcal{V}$, U_i computes (X_i', Y_i') // *(X_i', Y_i') is the point of intersection of the three circles centered at V_i, V_j, V_k and with distances: $d(U_i, V_i)$, $d(U_i, V_j)$ and $d(U_i, V_k)$.*
4: **if** there is a consensus on (X_i', Y_i') by the majority of triples **then**
5: U_i accepts the majority's position as its location: (X_i, Y_i).
6: **if** no (X_i', Y_i') can be calculated or the (X_i', Y_i') calculated according to a triple t is not the same as U_i's correct (majority) location (X_i, Y_i) **then**
7: U_i adds the sensors of this triple t to its *Liars* list
8: **else**
9: U_i adds the sensors of this triple t to its *CrossCheck* list
10: **end if**
11: **else**
12: U_i re-executes the algorithm once more from the beginning.
13: **end if**
14: U_i sends all its neighbors its location and its *CrossCheck* list.
15: As soon as a U_i receives a *CrossCheck* list from another $U_j \in \mathcal{U}$, this U_i checks the consistency between the two *CrossCheck* lists.
16: **if** A triple t is in both *CrossCheck* lists **then**
17: U_i puts the sensors of this triple t in its *TruthTellers* list.
18: **else**
19: U_i puts the sensors of this triple t in its *Liars* list.
20: **end if**

V. Correctness and Complexity Analysis

Lemma 2: Let n be the number of neighbor nodes of a location-unaware sensor U_i, l be the number of *liar*s among n. The *CrossCheck* list is correctly constructed by each U_i, if $n^3-3(2l+1)n^2+2(3l^2+6l+1)n-(2l^3+6l^2+4l)>0$.

Proof: Theorem 1 in paper [4] stated: Let n be the number of distance one neighbor nodes of a location-unaware sensor A. The execution of the majority rule in Algorithm *Majority-ThreeNeighborSignals* by A always gives a correct position in the presence of l *liar*s if inequality $n^3-3(2l+1)n^2+2(3l^2+6l+1)n-(2l^3+6l^2+4l)>0$ is satisfied. According to this theorem, each U_i should be able to calculate its position correctly, despite the presence of *liar*s.

When U_i gets its correct position, according to Line 7 in Algorithm 1, U_i compares its position to the coordinates obtained from a triple of sensors t. As stated in our assumption, U_i is able to distinguish two sets of coordinates. So, when these two sets of coordinates match, U_i puts this triple t into its *CrossCheck* list. ■

Lemma 3: Let U_i and U_j be two location-unaware sensors and V_i be a *liar*, and V_i's fake position be V_i'. $d(U_j,V_i)$ is the distance between U_j and V_i. $d(U_j,V_i')$ is the distance between U_j and V_i'. A second location-unaware sensor U_i can identify the *liar* V_i, when $d(U_i,V_j)\neq d(U_i,V_k)$.

Proof: Let $V3$ be one instantiation of V_i, $V3'$ be one instantiation of V_i', $d(U_i,V3)$ be the distance between U_i and $V3$, and $d(U_i,V3')$ be the distance between U_i to $V3'$.

As illustrated in Figure 2 and theorem 1, U_i is not able to tell if one of the triple t , node $V3$ is faking its position at $V3'$, if $V3'$ is on the circle centered at U_i with a radius $d(U_i,V3)$. This is because, $d(U_i,V3)=d(U_i,V3')$. See Figure 5[1].

Let (X_j,Y_j) be the coordinates of U_j. U_j can calculate its location (the value of (X_j,Y_j)) using the following three equations if $V1$, $V2$, $V3$ all tell their true position:

1) $(V1x-X_j)^2+(V1y-Y_j)^2=d(V1,U_j)^2$

2) $(V2x-X_j)^2+(V2y-Y_j)^2=d(V2,U_j)^2$

3) $(V3x-X_j)^2+(V3y-Y_j)^2=d(V3,U_j)^2$

Now we consider the case when $V3$ sends U_j its fake location $V3'$: $(V3'x,V3'y)$. The equations U_j uses to calculate the location (X_j',Y_j') are changed into:

1) $(V1x-X_j)^2+(V1y-Y_j)^2=d(V1,U_j)^2$

2) $(V2x-X_j)^2+(V2y-Y_j)^2=d(V2,U_j)^2$

[1]In Figures 5 and 6, $V3$ is one instantiation of V_i and $V3'$ is one instantiation of V_i'.

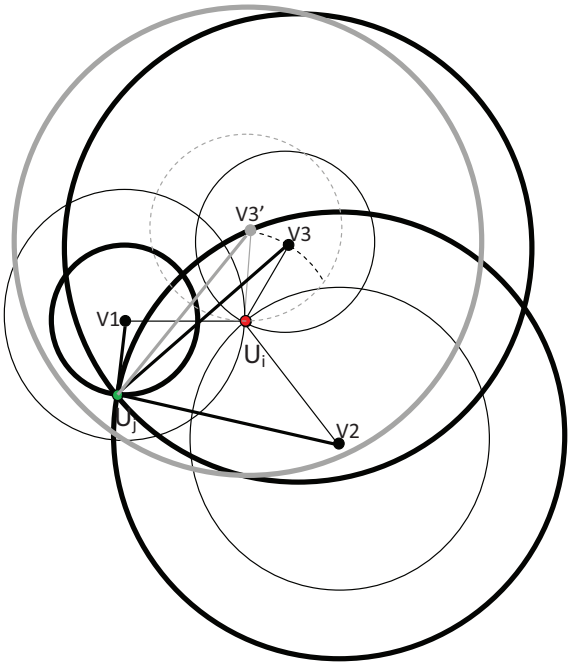

Figure 5. U_j is used in crosschecking with U_i in order to detect *liar* $V3$.

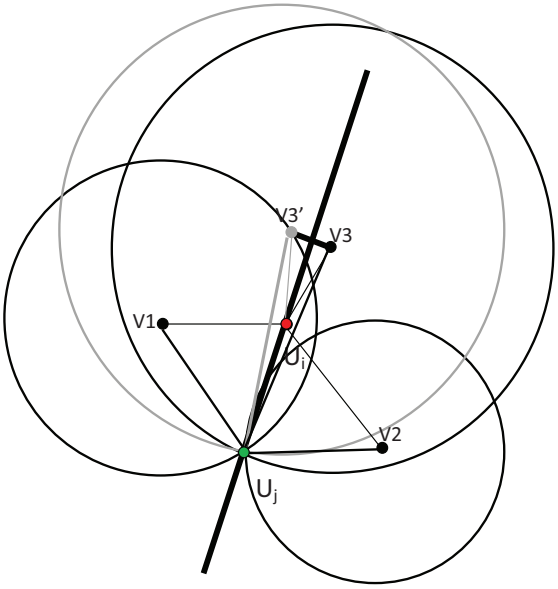

Figure 6. Case where U_j is located on the line that is orthogonal to the line passing through the real position and the fake position of a node (respectively $V3$ and $V3'$).

3) $(V3'x-X_j)^2+(V3'y-Y_j)^2=d(V3',U_j)^2$

If $d(U_i,V_j)=d(U_i,V_k)$ (see Figure 6), that is, U_j is positioned on the line that is orthogonal to the line passing through node $V3$ and its fake position $V3'$, after calculation, U_j gets coordinates (X_j',Y_j') and $(X_j',Y_j')=(X_j,Y_j)$. Because we assumed that $d(U_i,V_j)\neq d(U_i,V_k)$, it is clear

that $(X'_j, Y'_j) \neq (X_j, Y_j)$. This result is also illustrated in Figure 5.

■

Theorem 4: Algorithm *Cross Check* detects the triple t that contains *liar*s correctly.

Proof: According to Lemma 2, both U_i and U_j can construct their *CrossCheck* list correctly. After they have finished constructing their *CrossCheck* list, they send their *CrossCheck* lists to all the other sensor nodes (see Line 15 in Algorithm 1). We assumed that there exists at least one other location-unaware sensor U_j. U_j is not positioned on a line $\mathcal{L}$ that is orthogonal to the line passing through node V_i and its fake position V'_i. As proved in Lemma 3, one or more *liar*(s) in a triple t leads to either an incorrect position of U_i or failure for U_i to calculate its position. Also, because we assumed that each sensor is able to distinguish two coordinates, then it is clear that a triple t with *liar*(s) is noticed by either U_i or U_j. As described from Lines 16 to 20 in Algorithm 1, each location-unaware sensor node puts the sensors of triple t into its *liar* list when this triple t belongs to only its own *CrossCheck* list. We observe that a triple t either belongs to only one *CrossCheck* list, or it belongs to other *CrossCheck* lists. In the first case, it contains at least one *liar*, and in the second case, it contains only *truth tellers*. It is important to notice that each $U_i \in \mathcal{U}$ can conclude that a triple t does not contain any *liar* as soon as this U_i sees this triple t in its own *CrossCheck* list and one *CrossCheck* list it receives.

■

Theorem 5: Using Algorithm *Cross Check*, $O(n^2)$ messages suffice to correctly detect that the triple t contains *liar*s.

Proof: Initially, k *location unaware* sensor nodes send position requests to all the other $n-1$ nodes in the area (or neighborhood). This leads to maximum kn messages. Then $n-k$ sensors that are aware of their position, broadcast their position message. This step generates n^2 messages in the worst case. After a *location unaware* sensor node U_i constructs a *CrossCheck* list, it is going to broadcast its *CrossCheck* list so that another *location unaware* sensor node U_j can use this information to detect more *liar* triples. This step generates kn messages. The total messages adds up to $2kn+n^2$. Hence, using Algorithm *Cross Check*, $O(n^2)$ messages suffice to locate all the location-unaware sensors and correctly detect all the triples t that contain *liar*s.

■

VI. Conclusion

We have presented an attack on the localization algorithms of Alfaro et al. [3], [4] called the *evil ring attack*. This attack enables other attacks on routing protocols requiring node position information. We have formally demonstrated how to run the attack. We have also proposed an algorithm that detects the *evil ring attack*. The correctness of the algorithm has been demonstrated. Its complexity has been analyzed.

Acknowledgment

Financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC) is graciously acknowledged.

References

[1] M. Barbeau and E. Kranakis, *Principles of Ad Hoc Networking*, Wiley and Sons Ltd. 2007.

[2] B. Karp and H.T. Kung, *GPSR: greedy perimeter stateless routing for wireless networks*, Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom 2000), Boston, Massachusetts, United States, 2000, pp. 243-254.

[3] J. G. Alfaro, M. Barbeau, and E. Kranakis, *Secure Localization of Nodes inWireless Sensor Networks with Limited Number of Truth Tellers*, The 7^{th} Annual Communication Networks and Services Research (CNSR) Conference, IEEE Communications Society, Moncton, New Brunswick, Canada, pp. 86-93, 2009.

[4] J. G. Alfaro, M. Barbeau, and E. Kranakis, *Positioning of Wireless Sensor Nodes in the Presence of Liars*, Technical Report TR-10-04, School of Computer Science, Carleton University, March 2010.

[5] S. Capkun and J. P. Hubaux. *Secure positioning in wireless networks*, IEEE Journal on Selected Areas in Communications: Special Issue on Security in Wireless Ad Hoc Networks, 24(2):221-232, 2006.

[6] J. Hwang, T. He, and Y. Kim. *Secure localization with phantom node detection*, Ad Hoc Networks, 6(7):1031-1050, Elsevier, 2008.

[7] L. Lazos, R. Poovendran, S. Capkun. *ROPE: Robust position estimation in wireless sensor networks*, The 4^{th} Intl symposium on Information processing in sensor networks, 2005, p. 43.

[8] S. Delaet, P. Mandal, M. Rokicki and S. Tixeuil, *Deterministic secure positioning in wireless sensor networks*, IEEE International Conference on Distributed Computing in Sensor Networks (DCOSS'08), Lecture Notes in Computer Science, Springer Berlin/Heidelberg, Volume 5067, 2008, pp. 469-477.