Resilient Secure Localization and Detection of Colluding Attackers in WSNs

Wei Shi[1], Meng Yao[2], and Jean-Pierre Corriveau[2]

[1] University of Ontario Institute of Technology, Oshawa, L1H 7K4, Canada
wei.shi@uoit.ca
[2] Carleton University, Ottawa K1S5B6, Canada
mengyao86@gmail.com, jeanpier@scs.carleton.ca

Abstract. There exists extensive work in wireless sensor networks (WSNs) on security measures that guarantee the correctness of the estimation of the position of a node despite attacks from adversaries. But very little work has investigated how colluding attackers can modify the behavior of known attacks or even create new ones. In this paper, we first present an attack model that allows three types of colluding attackers to threaten the secure localization process and/or the attacker detection process. We then describe a decentralized algorithm that is used to determine the position of a location-unknown sensor U despite the presence of colluding attackers (that can alter any type of information being exchanged in a WSN in order to form an attack jointly). Most importantly, the proposed algorithm allows U to detect such colluding attackers in its sensing range. Our simulation results show that in both a uniformly deployed WSN environment and in a randomly deployed one, our Super Cross Check algorithm can achieve a high success rate for both secure localization and detection of colluding attackers.

Keywords: Secure Localization, Colluding Attacker Detection, Wireless Sensor Networks.

1 Introduction

1.1 Background

Wireless sensor networks (WSNs) have enabled a new form of communication between tiny embedded devices equipped with sensing capabilities. Such sensors act as the nodes of an ad-hoc network in which communication relies on a distributed collaborative exchange of information. Applications for WSNs range from environmental and health monitoring, to home networking and tracking systems (for objects, animals, humans, and vehicles). And in many WSNs applications, the positions of unknown (or equivalently, location-unknown) nodes play a critical role. Moreover, many fundamental techniques in WSNs (such as geographical routing, geographic key distribution, and location-based authentication) require determining the positions of unknown nodes. When a WSN is deployed in an unattended and/or hostile environment, it is vulnerable to threats and risks. Many attacks (such as wormhole, sinkhole and sybil ones) make the estimated positions incorrect. Such incorrect positions may have severe consequences in many applications. For example, a battlefield surveillance system incorrectly reporting enemy movement or wrongly identifying an ally as an enemy; a patient monitoring system sending the wrong location of a patient in critical condition; a forest fire monitoring system incorrectly reporting the location of a fire; a nuclear reactor monitoring system locating erroneously a malfunction. Thus, a secure localization scheme, that is, one that guarantees the accuracy of computed locations, is absolutely required.

Usually, there are two steps in a localization process: information acquisition and position calculation. Most adversaries attack the first step of a localization process. An adversary can either a) corrupt normal nodes into sending false localization information, or b) pretend to be a legitimate node in order to forge, alter or replay communication data. Such attacks will lead to inaccurate localization calculations (regardless of whether it is a centralized authority node that calculates the location of a location-unknown node, or such a node calculates its own location locally). Consequently, security measures have been extensively studied in order to make estimated positions correct despite attacks from an adversary. But several questions remain, in particular: a) What happens when several adversaries collude? and b) Can the attack model change and, if so, what kind of damage can it inflict on the localization process?

1.2 Related Work

Meadows et al. [1] analyze existing techniques for collusion prevention, and show how these techniques are inadequate for addressing the issue of collusion in sensor networks. Wang et al. [2] propose a novel localization algorithm called TMCA. This is a distributed algorithm based on the cooperation of non-beacon neighbor nodes. It is robust against some known attacks such as the wormhole, sybil and replay attacks. Even when there are more malicious anchor nodes than benign anchor nodes in a WSN, TMCA can still generate adequate localization results. The algorithm calculates an unknown node S's location using a distance bounding technique when S receives the coordinates (x_i, y_i) and distance d_i from a reference beacon. The Maximum Likelihood Estimate technique is used to receive a reasonably precise location of S. However, despite the algorithm being called "Tolerant Majority Colluding Attacks", there is no evidence of collaboration (e.g., exchange of messages) between attackers to form an attack jointly.

In [3], Garcia-Alfaro et al. introduce algorithms that enable the unknown nodes to determine their positions in the presence of neighbor sensors that may lie about their locations. In algorithm *Majority-Three Neighbor Signals*, all the neighbor anchor nodes of a sensor advertise their locations. For every three anchor nodes, the unknown node uses trilateration [4] to calculate a position. Then, a majority decision rule is used to obtain the final position of the unknown node. All triplets that compute a location different from this final one have their nodes considered to be liars. In [5,6], an Evil Ring (ER) attack is introduced. An evil ring attacker who lies about its position can successfully fool all the sensors that use trilateration to obtain or verify their locations. Algorithm *Cross Check* is presented to detect such attackers. The evil ring is an attack on the location determination algorithms of Garcia-Alfaro et al. When inquired, an attacker returns a fake location sitting on a circle centered at the victims location and with radius equal to the attacker-victim separation distance. The calculation of the distance between the victim and the attacker is not affected. A location-unknown node correctly determines its location. The attack, however, misleads such as node in getting and using wrong locations for its malicious neighbors. An evil ring attacker who lies about its position can successfully fool all the sensors that use trilateration to obtain or verify their locations. Algorithm *Cross Check* is presented to detect such attackers. Its main steps are:

- Request locations: Location-unknown node U sends using broadcast a location request to all the other nodes in the neighborhood.
- Calculate location: Using every possible three neighbor combination and their distances, node U calculates a location (x, y) according to the majority decision rule.
- Build a *cross-check* list: All neighbors in triplets in agreement with the majority are added to the *cross-check* list. The accepted location and *cross-check* list are sent using broadcast to neighbors.
- Liar detection: Node U waits until it receives two *cross-check* lists from two different neighbors. Every node present with identical location in all three *cross-check* lists is added to the neighbor table. Otherwise, it is added to the list of liars.

Most importantly, however, both [3] and [5,6] assume a dense network in which no sensor collusion exists.

In “Collaborative Collusion” [7], the CCAM model is proposed. In that model all malicious nodes can collaborate with each other to alter the location information they receive and/or jointly forward it. The authors present a solution to detect such malicious nodes. However their algorithm TSFD cannot detect the ER attackers introduced in [5,6]. Also, their proposed solution rests on the existence of a trusted base station that periodically broadcasts trusted grids to all nodes. In fact, attacker detection is performed only by this base station. Such calculation at a central node has several drawbacks. First, in order to forward the location information to a central node, a route to the latter must be known. This implies the use of a non-location-based routing protocol, which entails an additional communication cost. Second, because of the large volume of traffic to and from the central node, the battery lifetime of the nodes around the central node will be seriously impacted. Third, centralization hinders the robustness of the system: if the routes to the central node are broken, the nodes will not be able to communicate their location information to the central node and vice versa. In summary, a centralized implementation will not only reduce a network’s lifetime, but it will also increase its complexity and compromise its robustness. On the other hand, if location estimation takes place at each node, in a distributed manner, such problems can be alleviated [8].

1.3 Our Contribution

Very little work has been done on investigating how colluding attackers can change the behavior of known attacks or even create new attacks. Depending on the nature of each secure localization algorithm, sensors could collaboratively elude the location-unknown sensors by: 1) jointly announcing false information or 2) forming an illegal position (physical) pattern (e.g., more than two sensors are collinear). Either one of these two scenarios could lead to an erroneous calculation of a position. Furthermore, beyond location information, the colluding attackers can also alter other information or signals in the WSN under attack. For example, reputation lists are used in [2, 5, 6, 9, 10] in order to support different secure locational algorithms. In [2,5,6], the consistency of reputation lists is checked in order to detect malicious sensor nodes. All existing solutions assume that such reputation lists are not corrupted and that there are no colluding attackers that can jointly compromise this consistency verification procedure. In contrast, in this paper, we present a decentralized algorithm that is used to determine the position of a location-unknown sensor U when there are colluding attackers that can alter all types of information being exchanged in the WSN in order to form attacks jointly. Most importantly, the proposed algorithm allows U to detect such colluding attackers in its sensing range.

2 Colluding Attackers Detection Algorithm: Super Cross Check (SCC)

2.1 Attack Model and Assumptions

Let $\mathcal{K}$ be a set of location-known sensor nodes. Let $\mathcal{A}$ be a set of Anchor nodes, which know their own positions beforehand by either using GPS or being manually configuration [11, 12]. And let $\mathcal{S}$ be a set of regular sensor nodes, where $\mathcal{S} \subset \mathcal{K}$ and $\mathcal{A} \subset \mathcal{K}$. $\mathcal{U}$ denotes a set of location-unknown sensor nodes. Each location-unknown sensor $U \in \mathcal{U}$ can measure accurately the distance between itself and any other node in its sensing range. All sensor nodes are deployed on a $2-$dimensional plane $\mathcal{G}$. In $\mathcal{K}$, there are sensors (hereafter called *liars*), including Evil Ring attackers [5, 6], that can lie about their locations and any other information being exchanged with neighboring nodes in a liar's sensing range. Such lying behavior may be the result of malicious attacks or an unintentional act due to a sensor's physical malfunctioning. There are upper bounds on the number of tolerable liars, otherwise the algorithms in [3, 5, 6] fail. As a function of the liar number, Table 1 lists the minimum number of neighbors required to determine a location. We call a *liar* $C_i \in \mathcal{C}, \mathcal{C} \subset \mathcal{K}$ a *Colluding Attacker* if C_i and one or more other *liar(s)* jointly form an attack. A *Colluding Attacker* can be either a regular node or an Anchor node.

In this paper, we present a solution that, despite the presence of colluding attackers, allows a location-unknown sensor U to obtain its position and detect such attackers within its sensing range. In order to detect colluding attackers, we must know what kind of threatening behavior (affecting the accuracy of the secure localization process) sensor collusion can bring into the WSN. Clearly, the less information being exchanged between sensor nodes, the less chance adversaries have to attack successfully. In our solution the only messages being exchanged between sensor nodes are the coordinates of each sensor and its *Cross Check Lists* (hereafter *CC* lists). In this paper, we do not focus on a colluder's ability to attack the system by corrupting periodic 'Hello' messages (used to check if neighbors are alive and to exchange routing information).

We organize colluding attackers into three categories. Attacks from attackers in all three categories involve, in part, sending out an altered *CC* list. Beyond having this common behavior, further categorization depends on how an attacker lies about its location during the localization process:

1. a liar lies about its location by using a randomly generated fake location;
2. a liar lies about its location by giving out a fake location: in cooperation with two other attackers, it returns the location of a location-unknown sensor $U \in \mathcal{U}$.
3. a liar lies about its location by giving out a fake location that sits on a circle centered at the victim's location and of a radius corresponding to the distance between the attacker and victim. The attack succeeds and is undetectable by any existing *liar* detection algorithm, except for the one presented in [5,6]. Namely, only algorithm *Cross Check* can detect such an *Evil Ring* attack. But this type of colluding attackers, like in the other categories, also send out colluded *CC* lists, which algorithm *Cross Check* cannot address.

In the rest of this paper, we will focus on explaining how to deal with the third category of liars. This is because an algorithm that can detect such liars (of our third category) can *de facto* handle the first two categories of liars. These first two categories of attackers must still be identified because they use different attacking behaviors to jeopardize the localization process, and such differences must be simulated.

Also, we will compare our proposed solution only with the algorithm Cross Check presented in [5,6] since only that algorithm can detect non-colluding liars similar to those of our third category.

Table 1. Minimum number of location-aware neighbors required to determine a correct location, as a function of the number of liars [3]

Number of Liars	Min Number of Neighbors
1	7
2	11
3	16
4	21
5	26
10	31
15	74
20	98

We postulate that in order for a location-unknown node U to systematically detect a third category colluding attacker, the number of non-colluding attackers in the intersection of the sensing circles of U and A should be at least two more than the number of colluding attackers in U's sensing range.

2.2 SCC Algorithm

There are two steps in this algorithm:

- obtain the position of a location-known sensor $k \in \mathcal{K}$
- detect the colluding attackers.

Step one: calculate the position and create a CC list for each U. In this first step (Lines 1-9 in Algorithm 1 below), we use the trilateration technique to calculate the location of each U. We then used a majority decision rule, as used in [5,6], to obtain the final position of the unknown nodes.

Step 2: exchange CC lists and detect the colluding attackers. In this second step (Lines 10-29 in Algorithm 1), upon receiving a request for its CC list, each $k \in \mathcal{K}$ sends out its CC list. If it does not have a CC list, it will construct one the same way a location-unknown sensor U does. Naturally, if k is a colluding attacker, it will send out a CC list that does not reflect its real calculation results. For example, it may purposely delete a few sensors that should have been in the list, in order to give the illusion that these deleted sensors could be colluding attackers. This attack could succeed if there were several such colluding attackers lying about the same fact. This second step of the algorithm detects the three categories of colluding attackers by using a voting technique: U requests a CC list from its neighbor nodes. Upon receiving a CC list L_i, U gives a positive credit to a node $k \in K$ if this node (that is, its coordinates) is in both U's CC list and L_i, a negative credit to k otherwise. Once U receives all the CC lists, it will compute the number of positive and negative credits of each neighboring node. If a neighboring node k received two more positive credits than negative credits, U can conclude that k is not a colluding attacker and U can use k to construct its routing table. Otherwise k is identified as a colluding attacker.

The pseudo code for this algorithm is shown in Algorithm 1.

3 Simulation and Evaluation

In this section, simulation results are presented and analyzed. The simulations are performed using Omnet 4++. We conduct tests under the following two scenarios: 1) uniformly deployed Anchor nodes and regular sensor nodes and 2) randomly deployed sensor nodes. We evaluate the performance of our proposed algorithm by measuring the success rate for the detection of colluding attackers and by comparing our results with those of the Cross Check algorithm presented in [6]. We explain the details of these two scenarios separately.

Algorithm 1. Super Cross Check

```
1: repeat
2:     Request neighbors' locations.
3:     for all triplets of neighbors ($V_1$; $V_2$; $V_3$) do Compute the intersection point of
   the three circles centered at $V_1$; $V_2$; $V_3$ with radius $d_1$, $d_2$, $d_3$.
4:     end for
5: until there is a consensus on $(x, y)$ determined by the majority of triplets.
6: Accept $(x, y)$ as the location of $U$.
7: for all triplets of neighbors ($V_1$; $V_2$; $V_3$) in agreement with the majority do
8:     Add the locations $V_1$; $V_2$; $V_3$ to $U$'s $CC$ list.
9: end for
10: Broadcast location of $U$ and its $CC$ list.
11: request $CC$ lists from all the neighbors that are in $U$'s $CC$ list
12: for each neighbor $i$ on $U$'s $CC$ list (referred to as $I$) do
13:     Select all sensors in the intersection of the two sensing circles of $U$ and $i$
14:     for all the selected sensors (referred to as $c \in C$) do
15:         compare the $CC$ list from $c$ with the one from $U$
16:         if $i$ is in both $CC$ lists then
17:             give a positive credit to $i$
18:         else if $i$ is not in the $CC$ list received from $i$ then
19:             give a negative credit for $i$
20:         end if
21:     end for
22: end for
23: for each $i \in I$ do
24:     if $i$ received 2 more positive credits than negative credits then
25:         this node is not a colluding attacker and it can be used to construct $U$'s
    routing table
26:     else
27:         this node is a colluding attacker
28:     end if
29: end for
```

3.1 Uniform Sensor Deployment

In this section, we consider the situation in which all sensors are uniformly deployed in a WSN. In Figure 1, solid black dots represent Anchor nodes, which have $\mathcal{R}_a = 2d$ with $(d > 0, d = \overline{AB})$ as their sensing range; and grey dots represent the regular location-known sensors, which have $\mathcal{R}_r = \sqrt{2}d$ with $(d > 0)$ as their sensing range. Any of these Anchor nodes and regular sensor nodes could be colluding attackers. Additionally, each black and white dot represents a location-unknown sensor node that also has $\mathcal{R}_r = \sqrt{2}d$ with $(d > 0)$ as its sensing range. A key observation is that, in the worst case, all the colluding attackers in the WSN under attack belong to our third category.

As mentioned earlier, in order for a location-unknown node U to systematically detect a third category colluding attacker A (or B) in Figure 1, the number of non-colluding attackers in the intersection of the sensing circles of U and A should be at least two more than the number of colluding attackers in U's

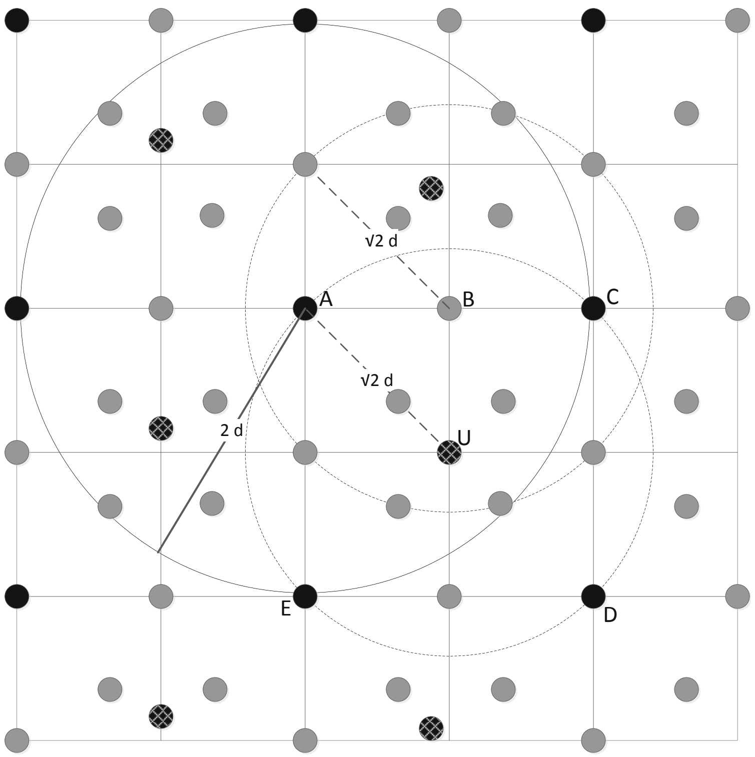

Fig. 1. Uniform deployment

sensing range. In our proposed uniform deployment WSN, this assumption leads to approximately 33.3% of third category colluding attackers in the total number of location-known sensor nodes. We focus here on the success rate of detecting third category colluding attackers after executing algorithm Super Cross Check and then compare these results against the ones of algorithm Cross Check. Our simulation results are presented in Figure 2. The solid line shows that in the proposed uniformly deployed WSN, the success rate of detecting colluding attackers (which will all be third category colluding attackers in the worst case) after executing algorithm Super Cross Check is 100% consistently, when the percentage of colluding attackers does not exceed 33%. We also observe that in order to keep a 100% colluding attacker detection success rate, the percentage of third category colluding attackers among all the colluding attackers should be inversely proportional to the percentage of colluding attackers among all location-known sensor nodes. In other words, beyond 33% of third category colluding attackers, having a higher percentage of colluding attackers from the two first categories among all the location-known sensor nodes does not affect the colluding attacker detection success rate. The dashed line in the figure corresponds to the performance of algorithm Cross Check under the same setup. The results of algorithm Cross Check show that in each cell of the grid (e.g. cell *ACDE* in Figure 1), as long as

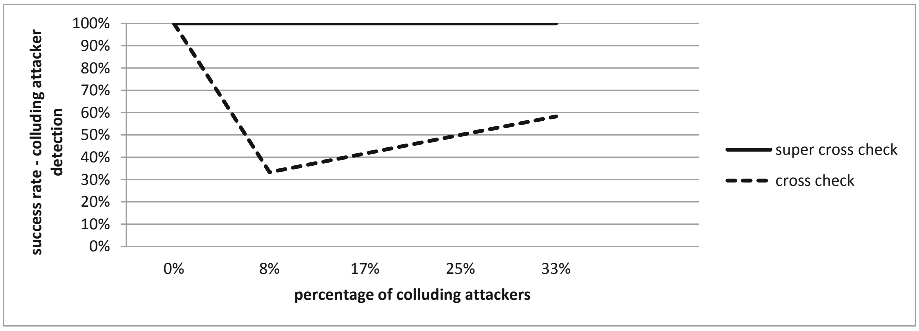

Fig. 2. Comparison of colluding attacker detection success rate between algorithms Super Cross Check and Cross Check in a uniformly deployed WSN

there are more than 1 colluding attackers, the success rate of detecting colluding attackers will drop drastically (to as low as 30%). Interestingly, we observe that when the percentage of colluding attackers among all location-known sensor nodes is between 8% to 33%, the success rate increases. However, after careful analysis, we conclude this increase does not correlate to the ability of algorithm Cross Check to detect colluding attackers. Instead, this rate increase stems from the fact that the algorithm will wrongly label some nodes as colluding attackers. Thus, the more genuine colluding attackers present in the WSN, the more nodes labeled arbitrarily by algorithm Cross Check as colluding attackers will end up being real colluding attackers, thus increasing the detection rate.

3.2 Random Sensor Deployment

In this section, we consider the situation in which all sensors are randomly deployed in a WSN. We compare the performance of algorithm Super Cross Check and algorithm Cross Check with respect to the following two aspects: the success rate of localizing location-unknown sensors and the success rate of detecting colluding attackers.

Figure 3 shows the success rate of detecting colluding attackers using algorithm Super Cross Check in a randomly deployed WSN in which there are 80 non-colluding attackers and 10 location-unknown sensors. We can see from this figure that the success rate drops to 90% when the number of colluding attackers is around 50, which entails that 1 of the 10 location-unknown sensors under test did not receive its correct location. This is because in this random deployment setup, the number of colluding attackers exceeds the number of non-colluding attackers. Otherwise, algorithm Super Cross Check's success rate for secure localization of location-unknown sensor nodes in a randomly deployed WSN is 100%.

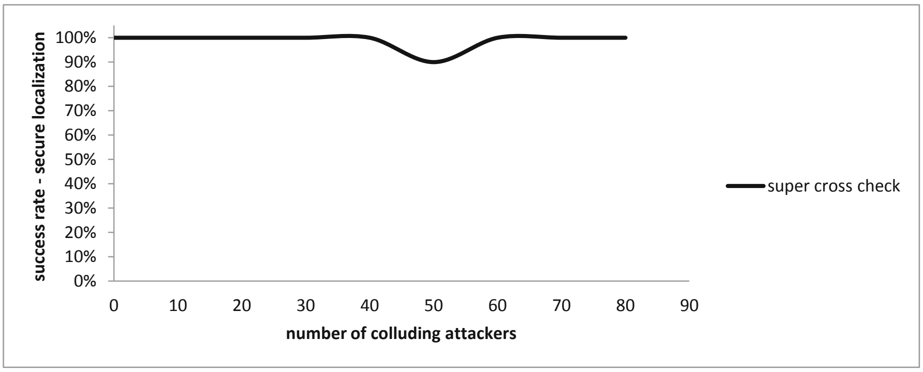

Fig. 3. Secure localization success rate of algorithm Super Cross Check in a randomly deployed WSN

Figure 4 illustrates the success rate of detecting colluding attackers using algorithm Super Cross Check, when there are 80 non-attacking nodes (location-known sensors) and 10 location-unknown nodes in the WSN. We let the ratio of the three categories of colluding attackers be 3 : 4 : 3 and the total number of colluding attackers increase from 0 to 80. The results show that algorithm Super Cross Check's success rate at detecting colluding attackers is never lower than 93%, while the success rate at detecting colluding attackers of algorithm Cross Check varies between 22% to 42%.

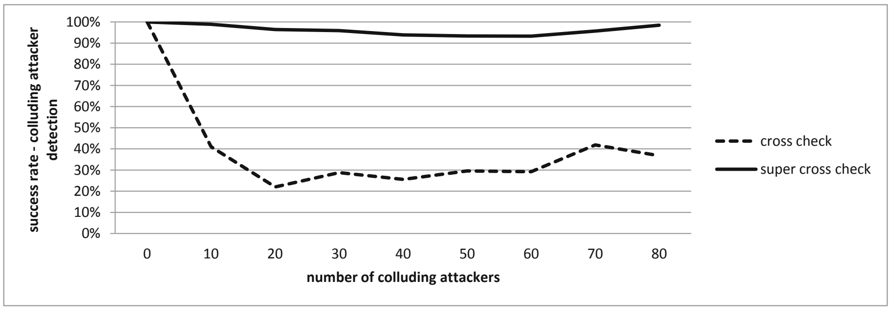

Fig. 4. Comparison of colluding attacker detection success rate between algorithms Super Cross Check and Cross Check in a randomly deployed WSN

4 Conclusions

When a WSN is deployed in unattended and/or hostile environments, it is vulnerable to threats and risks. Many attacks make the estimated positions incorrect.

Such incorrect positions may lead to severe consequences in many applications. Thus, security measures have been studied extensively in order to make the estimated positions correct despite the attacks from an adversary. But very little work has been done on investigating how colluding attackers can change the behavior of known attacks or even create new attacks. In this paper, we present an attack model that allows three types of colluding attackers to attack the secure localization process and/or the attacker detection process. We then present a decentralized algorithm that is used to determine the position of a location-unknown sensor U when there are colluding attackers that can alter all types of information being exchanged in the WSN in order to form attacks jointly. Most importantly, the proposed algorithm allows U to detect such colluding attackers in its sensing range. The simulation results show that in both uniformly deployed and randomly deployed WSN environments, our algorithm Super Cross Check achieves a significantly high success rate for both secure localization and colluding attackers detection.

References

1. Meadows, C., Poovendran, R., Pavlovic, D., Chang, L., Syverson, P.: Distance bounding protocols: Authentication logic analysis and collusion attacks. In: Secure Localization and Time Synchronization in Wireless Ad Hoc and Sensor Networks. Springer (2007)
2. Wang, X., Qian, L., Jian, H.: Tolerant majority colluding attacks for secure localization in wireless sensor networks. In: 5th International Conference on Wireless Communications, Networking and Mobile Computing, pp. 1–5 (2009)
3. Garcia-Alfaro, J., Barbeau, M., Kranakis, E.: Secure geolocalization of wireless sensor nodes in the presence of misbehaving anchor nodes. In: Annals of Telecommunications, pp. 1–18. Springer (2011), doi: 10.1007/s12243-010-0221-z
4. Niculescu, D., Nath, B.: Ad hoc positioning system (APS). In: The 2001 IEEE Global Telecommunications Conference of the IEEE Communications Society, pp. 2926–2931 (2001)
5. Shi, W., Barbeau, M., Garcia-Alfaro, J., Corriveau, J.-P.: Detection of the Evil Ring Attack in Wireless Sensor Networks Using Cross Verification. In: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2010), Montreal, Canada, 8 pages (June 2010)
6. Shi, W., Barbeau, M., Garcia-Alfaro, J., Corriveau, J.-P.: Handling the Evil Ring Attack on Localization and Routing in Wireless Sensor Networks. Journal of Ad Hoc & Sensor Wireless Networks (to appear, 2012)
7. Jiang, J., Han, G., Shu, L., Chao, H., Nishio, S.: A novel secure localization scheme against collaborative collusion in wireless sensor networks. In: 7th International Wireless Communications & Mobile Computing Conference, pp. 308–313 (July 2011)
8. Savvides, A., Han, C., Strivastava, M.B.: Dynamic fine-grained localization in Ad-Hoc networks of sensors. In: 7th Annual International Conference on Mobile Computing and Networking (MobiCom 2001), pp. 166–179. ACM, New York (2001)
9. Liu, D.G., Ning, P., Du, W.: Detecting malicious beacon nodes for secure location discovery in wireless sensor networks. In: 25th Int. Conf. on Distributed Computing Systems (ICDCS), pp. 609–691. IEEE Computer Society Press, Washington (2005)

10. Srinivasan, A., Wu, J., Teitelbaum, J.: Distributed reputation-based secure localization in sensor networks. Journal of Autonomic and Trusted Computing (2007)
11. Du, Q., Qian, Z., Jiang, H., Wang, S.: Localization of Anchor Nodes for Wireless Sensor Networks. In: New Technologies, Mobility and Security (NTMS 2008), pp. 1–5 (2008)
12. Tian, S., Zhang, X., Wang, X., Sun, P., Zhang, H.: A Selective Anchor Node Localization Algorithm for Wireless Sensor Networks. In: International Conference on Convergence Information Technology, pp. 358–362 (2007)