Wormhole Detection in Wireless Sensor Networks Using Spanning Trees

Károly Harsányi
Machine Perception Laboratory
Hungarian Academy of Sciences,
Institute for Computer Science and Control
(MTA SZTAKI)
Budapest, Hungary
harsanyika@sztaki.mta.hu

Attila Kiss
Machine Perception Laboratory
Hungarian Academy of Sciences,
Institute for Computer Science and Control
(MTA SZTAKI)
Budapest, Hungary
attila.kiss@sztaki.mta.hu

Tamás Szirányi
Machine Perception Laboratory
Hungarian Academy of Sciences,
Institute for Computer Science and Control
(MTA SZTAKI)
Budapest, Hungary
sziranyi@sztaki.mta.hu

***Abstract*—Wireless sensor networks and ad-hoc networks are gaining popularity rapidly due to their ability to solve challenging problems and the fact that thanks to recent technological advancements it is now possible to build smarter and denser networks. For example, they serve as the basis of the Internet of Things. Naturally, it is in the users best interest to develop increasingly secure networks. In some cases, the sensors are used in unknown or hostile environments. This and the vulnerability of the wireless communication channels used by arbitrary mobile communication networks means that they are exposed to various kinds of attacks. One of the most severe threats is the wormhole attack because an adversary can start the attack without compromising sensors or breaking through cryptographic defense mechanisms. In this paper, we propose a novel method for detecting wormhole attacks and identifying the affected sensors. Our approach does not rely on using any special measurement, only the connectivity information of the network.**

***Index Terms*—sensor networks, wormhole, security, network theory**

I. Introduction

Recent developments in the fields of wireless communication and distributed processing led to a rapid increase in the use of wireless and ad-hoc sensor networks. Today they are widely used with a variety of applications in surveillance, environmental monitoring, security and military technologies. The sensors rely on some kind of wireless communication and the networks are often set up in an open environment that is usually unexplored or hostile. This makes security and privacy an enormous challenge.

One of the most serious security threats is the so called wormhole attack, first defined in [1], [2] and [3]. An adversary can perform this kind of attack with limited resources: without compromising any sensors in the network or bypassing any cryptographic defense. In order to launch the attack, the intruder places two radio receivers connected by a high-speed, high-capacity channel in distant parts of the network. Signals captured by the receivers are sent through the 'wormhole link' to the other endpoint and there they are replayed (See figure 1). This radically changes the network topology and connectivity: it causes the sensor nodes around the receivers to recognize each other as direct neighbors and as a result they communicate through these wormhole links. This makes connectivity-based localization algorithms unreliable, causing any application that rely on geographic information to be deceptive (See figure 2). Furthermore, the wormhole tunnel draws a large amount of data traffic, so the attacker can launch other kinds of attacks, such as selectively modifying and dropping packets sent through the wormhole link or forwarding them out of order, etc. The adversary can also use the wormhole in a passive manner: eavesdropping, gathering packets and analyzing network traffic. Thus the wormhole attack can act as an instrument for orchestrating other more aggressive attacks to destroy and control different network protocols.

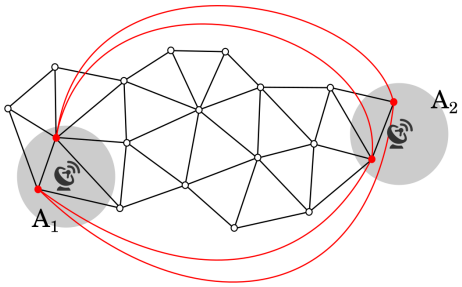
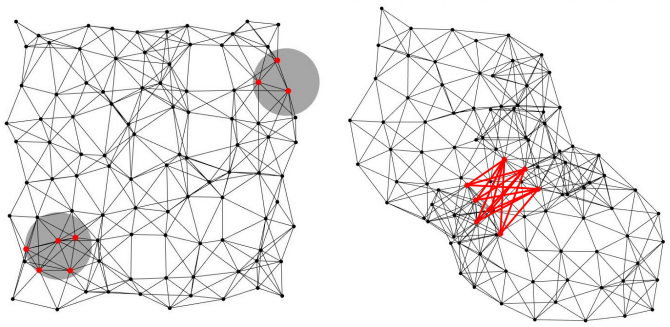


Fig. 1: Example of a wormhole attack. A_1 and A_2 denote the areas directly affected by the two receivers. As a result of the attack, new connections are established between all nodes in A_1 and all nodes in A_2.

In this paper, we introduce a new method for detecting wormhole attacks. Our idea is based on the fact that wormhole links often provide the new shortest paths between distant sensors in the network. This means that if we isolate a neighborhood of sensors that are directly affected by the wormhole, the change in the length of the shortest paths from an arbitrarily chosen sensor node and the rest of the nodes

This work was supported by the Hungarian Scientific Research Fund (No. OTKA/NKFIH 120499).



(a) (b)

Fig. 2: (a) shows a sensor network under a wormhole attack. The gray circles show the affected areas, the red dots mark the compromised sensor nodes. (b) shows the result of running the As-rigid-as-possible localization algorithm (ARAP) [4] on the network in the presence of the wormhole. The red lines show new connections introduced by the wormhole, and the red dots mark the influenced nodes. We can see that the wormhole connections mislead the localization algorithm, causing incorrect position estimates for most of the sensors.

will show a big variance. In other words, the paths to nodes that are closer to the farther endpoint of the wormhole will change, however, the rest of the paths will be unaffected.

The rest of this article is organized as follows. In Section 2 we discuss prior works. In Section 3 we introduce our approach in detail. In Section 4 we present simulation results and Section 5 contains our conclusions.

II. Related Work

In recent times the problem of detecting wormhole attacks received substantial attention. As a result, various protocols and countermeasures have been proposed.

A number of solutions aim to expose wormhole attacks with the help of distance or timing analysis. These methods attach the location of the sender node or some time information to the packets sent through the network. This way, after each received packet the sensors can verify it. If the transmission is not feasible according to the physical characteristics of the network, the presence of a wormhole is very likely. However, these approaches rely on extremely precise, synchronized clocks [3], [5], or they require special hardware such as GPS to obtain the geographical locations of the nodes [3], [6]. This leads to increasing hardware costs. Hu and Evans [7] propose another approach relying on special hardware. They use a cooperative protocol between the sensors with the help of directional antennas to identify false neighbors.

Some techniques [8]–[10] use special nodes called *guard nodes*. These nodes are aware of their geographical location and they benefit from higher transmit power and different antenna characteristics. Naturally, the dependence on guard nodes greatly limits the applicability of these methods.

Other approaches build on the assumption that at one point the network is attack free and the sensors can obtain valid connectivity information. Buttyán et al. [11] detect changes in the lengths of the shortest paths in the network in order to identify wormholes. Similarly, in [12] statistical analysis of the distortions in multi-path routing is used.

The method described in [13] uses connectivity information to look for 'forbidden structures' in the connectivity graph. However, it requires knowledge of the communication model and the node distribution. Otherwise, the algorithm shows a significant decline in precision.

Some other solutions rely solely on network topology and connectivity information. MDS-VOW [14] reconstructs the network layout with multi-dimensional scaling [15] and surface smoothing. Then it detects wormholes by identifying anonymities on the assembled network. However, this method runs in a centralized manner and only works in the cases when the wormhole attack inserts just a single false edge into the network. Dong et al. introduce WormCircle [16], a fully distributed connectivity based method. The idea is based on wave propagation. For every node, they examine its k-hop neighborhood. Generally, a subgraph like this has a circle shaped boundary, but in the presence of a wormhole the boundary forms two circles. WormCircle works well in networks with high density, but if the nodes have a small average degree WormCircle's detection rate drops significantly. Ban et al. [17] apply local connectivity tests on each node, on the premise that with the removal of wormhole it's neighborhood breaks down into multiple components. After they acquired a group of candidate nodes, they search for maximal complete bipartite graphs amongst these candidates using the algorithm from [18]. Similarly, [19] uses MDS-MAP [15] locally and checks for distortions in the network in order to obtain candidates. Afterwards, it uses the scheme from [18] as fine-graining to reduce the number of false positives. These last two methods work well for wormholes with large radius due to the fact that large wormholes introduce large complete bipartite subgraphs, but if the wormhole endpoints only directly influence a few sensors, they have a tendency to be unsuccessful.

III. Our Contribution

We introduced several state-of-art methods created to deal with the problem of wormhole attacks. However, all these methods have limitations. Many of them rely on special hardware, or special guard nodes. Some methods are based on the assumption that a wormhole only inserts a single false edge into the network. Others are solely reliable for wormholes that introduce large complete bipartite subgraphs.

Now, we present the detailed description of our approach. We use only the connectivity information of the network to find and isolate nodes under a wormhole attack. Our idea is based on the presumption that the removal of the wormhole edges causes sizable changes in the lengths of shortest paths between some of the nodes in the network, while other shortest paths remain unchanged. In order to monitor the changes, we run breadth-first searches from some selected nodes called 'root nodes' while we iteratively isolate other sensors and their neighborhoods.

Breadth-first search can be run in a distributed manner. The source node can send out a starting signal or packet containing its depth (0). Next, receiving nodes add 1 to the depth and forward the changed packet etc. After every node determines their distance, they can send it to the source node on the route designated by the now completed spanning tree. We can use this feature to make our algorithm almost completely decentralized. Although, the root nodes have to do some additional but trivial calculations (See Algorithm 2).

A. Selecting the root nodes

The number and distribution of the root nodes affects the accuracy of the algorithm. Choosing just one root node can be insufficient because there is no guarantee that we pick a node with the right position in the network. Root nodes that are directly affected by one of the endpoints will fail to detect it. Similarly, if a root node's distance is roughly the same from the two wormhole endpoints, or it is too far from them, we are less likely to observe the substantial changes caused by the removal of the wormhole. On the other hand, choosing too many roots can significantly increase the runtime of our algorithm. To tackle this problem we use Algorithm 1. First, we add every node to the set of possible root nodes. Then we iteratively run the following steps until this set is empty: We choose the node with the smallest ID as the first root. Next, this node floods the network with the message that it is now a root node and every node within k-hop distance will be removed from the set. It is easy to see that the parameter k introduces a trade-off. A small k leads to a large number of root points. This boosts the accuracy but also increases the runtime. The number of root nodes also depends on the density of the network. Let us denote the number of nodes in a network by n and the density of the graph by d. Based on our measurements we decided to choose $k = \frac{n}{4 \cdot d}$. This way our algorithm outperforms many of the other algorithms in runtime and efficiency.

Algorithm 1: Selecting roots for spanning trees

```
function get_roots (Con, L, k);
  Input : Con: the connectivity matrix of a network, L:
          the list of nodes, k: integer
  Output: R: a list containing the root points
S = empty set
add every node from L to S
R = empty list
while S is not empty do
    v = the element with the smallest ID from S
    add v to R
    remove v from S
    run a BFS with max depth k to obtain the nodes
      within k-hop distance from v
    N = nodes within k-hop distance from v
    S = S − N
end
return R
```

B. Identifying affected nodes

Algorithm 2: Finding wormhole nodes

```
function find_affected_nodes (Con, R, λ);
  Input : Con: the connectivity matrix of a network, R:
          the list of root nodes, λ: float
  Output: C: the list candidates
npoints = number of nodes
nroots = number of roots
VarMat = npoints × nroots dimensional matrix
  containing zeros
for Every r root node in R do
    D = npoints dimensional vector of zeros
    run a BFS from r and store the distances to all nodes
      in D
    F = list containing the direct neighbors of r and r
      itself
    for Every v node in F do
        VarMat[v, r] = −1
    end
    for Every v node not in F do
        Ddif = empty list
        D' = npoints dimensional vector of zeros
        Con' = a copy of Con
        T = list containing the direct neighbors of v and
          v itself
        remove every edge from Con' that connects
          nodes in T to the rest of the network
        run a BFS from r using Con' and store all the
          distances in D'
        for Every w node in the network do
            if D'[w] < infinity then
                append (D'[w] − D[w]) to Ddif
            end
        end
        VarMat[v, r] = variance of Ddif
    end
end
Avgs = npoints dimensional vector of zeros
for Every v node in the network do
    Avgs[v] = the average of all positive values in the
      vth row of the VarMat matrix
end
m = the mean of Avgs
for Every v node in the network do
    if Avgs[v] > m ∗ λ then
        append v to C
    end
end
return C
```

Once we determined which sensors will serve as roots, we can launch our detection algorithm. First, we create a matrix $VarMat$ with size $n \times r$ where n is the number of nodes and r is the number of roots. Then, for every root node r_i,

we repeat the following steps. We run a breadth-first search to determine the distance from r_i to every other node in the network, and we store these in a vector D_i, where $D_i[j]$ is the distance from r_i to the jth node.

Then for every node v, if v is not r_i or any of r_i's neighbors, we run the following test:

(1) we remove v and its neighborhood
(2) we run an other BFS to determine the distance of every node w in the modified graph
(3) if a w is not reachable from r_i we ignore it. Else, we add the $D'_{iv}[w] - D_i[w]$ to a container $Ddif_{iv}$, where $D'_{iv}[w]$ is the distance from r_i to w after removing v and its neighbors.

Finally, we measure the variance of $Ddif_{iv}$ and store it in the $VarMat$ matrix: $VarMat[i, v] = Var(Ddif_{iv})$. In this iteration we skipped the cases when $v = r_i$ or one of r_i's neighbors. We will examine these nodes from other roots. As an indicator we write -1 into $VarMat[i, v]$.

After we inspected every possible node from every root, we calculate the average of variances for every node v, that is the average of the positive values in the vth row of $VarMat$. Then, we compute the average of these averages: m. If the average variance for a node is higher than λm we add the node to the list of candidates. λ is the second parameter of our algorithm. It introduces a trade-off between the detection rate and the number of false positives. If we choose a larger λ we reduce the number of false positives, but we might fail to detect some wormholes. Algorithm 2 shows the pseudo-code for the algorithm described above.

After the algorithm is finished and we obtained the candidates nodes, we can further lower the number of false positives by examining the subgraph inducted by these nodes. If a node is isolated in this inducted graph, then it is certainly not a wormhole and we can remove it from the list of candidates. Finally, in order to shut off the wormhole, we turn off all the sensors in the candidate list, or prohibit every communication between them.

IV. Results

In this section, we evaluate our method under different circumstances, including different node deployments, communication models, and network density.

We use two deployment models: random placement and perturbed grid. In random placement, we choose the coordinates for the nodes uniformly and independently from a given area. In the perturbed grid model, we place the sensors on an $n \times m$ grid, and perturb them from their initial positions $(x \times y)$: their new coordinates will be selected uniformly from $[x - pd, x + pd] \times [y - pd, y + pd]$, where p is the displacement parameter and d is the length of the squares' sides in the original grid, in our tests we used $p = 0.75$. Deploying the network with random placement results in an uneven network with irregularities, while the perturbed grid is often used to simulate a manual deployment.

We apply unit disk graph (UDG) and quasi-UDG communication models to establish the connections in the network.

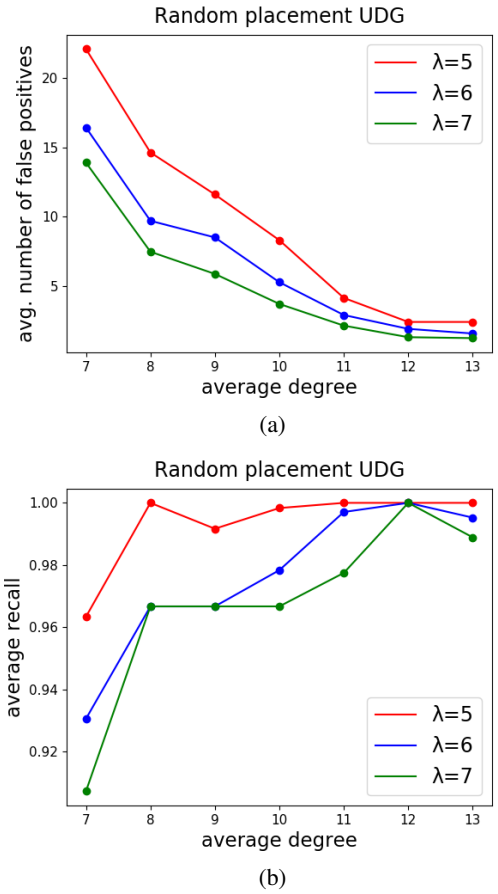
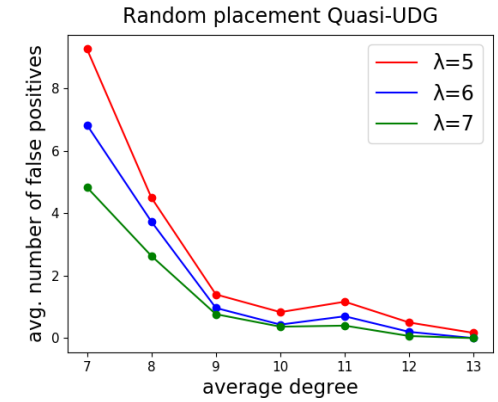


Fig. 3: Results on a network with 900 nodes, using UDG communication model and random placement. We adjusted the radius of the sensors the achieve different network densities. (a) shows the average number of false positives, and (b) displays the average recall. Each point corresponds to the average of 30 simulations with the same average degree. We ran our algorithm with $\lambda = 5, 6, 7$, to demonstrate how this parameter influences the outcome.

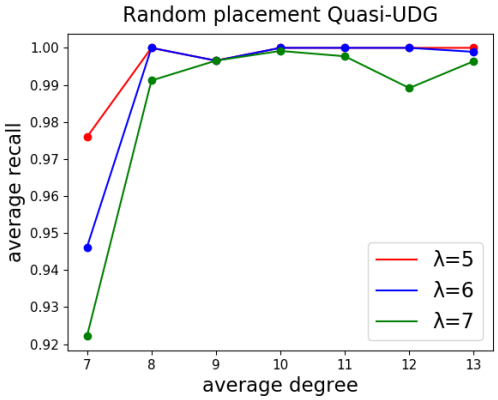
In UDG model, two nodes are connected if and only if their distance is shorter than the communication radius R. In the quasi-UDG model, there is a link between two nodes if their distance is shorter than r, and there is a link with some probability if their distance is between r and R, in our experiments we used $r = 0.5 * R$, and we adjusted R to obtain networks with various densities.

Another important factor is the distance between the centers of the examined wormhole. A wormhole with distant endpoints produces larger distortion for localization algorithms and it can draw more data traffic. A shorter wormhole causes less damage, but it is harder to detect. During our experiments, we focused on the detection of longer wormholes: we generated wormholes in such way that the hop distance between the two sets of wormhole nodes in the original network is at least 8.

Since the λ parameter in our algorithm offers a trade-off between the number of false positives and the detection rate,

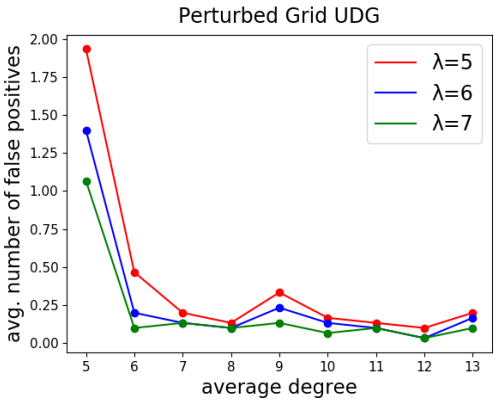


(a)

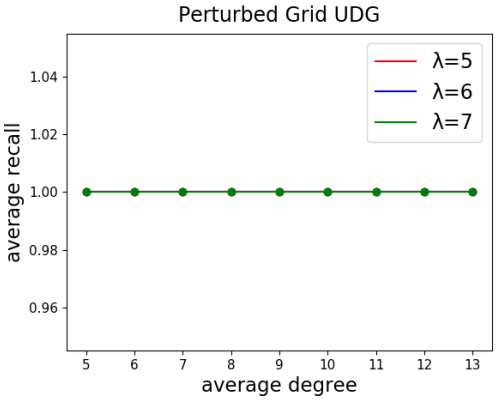


(b)

Fig. 4: Results on a network with 900 nodes, using Quasi-UDG communication model and random placement. We adjusted the radius of the sensors the achieve different network densities. (a) shows the average number of false positives, and (b) displays the average recall. Each point corresponds to the average of 30 simulations with the same average degree. We ran our algorithm with $\lambda = 5, 6, 7$, to demonstrate how this parameter influences the outcome.



(a)



(b)

Fig. 5: Results on a network with 900 nodes, using UDG communication model and perturbed grid deployment. We adjusted the radius of the sensors the achieve different network densities. (a) shows the average number of false positives, and (b) displays the average recall. Each point corresponds to the average of 30 simulations with the same average degree. We ran our algorithm with $\lambda = 5, 6, 7$, to demonstrate how this parameter influences the outcome.

we ran our tests with $\lambda = 5, 6, 7$. We generated 30 networks with 900 nodes for every deployment model, communication model and network density. We monitored how the algorithm performs under these circumstances by measuring the average number of false positives and the average recall over these test cases.

Figures 3, 4, 5, 6 show our results. The tests clearly demonstrate the effectiveness of our algorithm. It achieves close to 1.0 recall on networks with random deployment and 1.0 recall using perturbed grid. The number of false positives is relatively low especially for perturbed grid and for random placement with an average degree larger than 8.

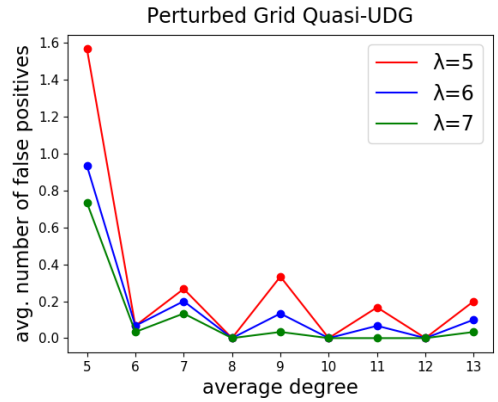
V. Conclusion

In this work, we introduced a novel approach for wormhole detection in sensor networks. Our approach does not rely on special hardware, special guard nodes or on statistics/information about the network prior to the attack. We use only the network's connectivity information. The algorithm runs in a distributed manner, the communication costs are roughly the same for every sensor in the network, and no costly calculations are required. Furthermore, the accuracy of the proposed algorithm is not affected by the number of wormhole nodes. We verified the effectiveness of our algorithm through vigorous tests in scenarios with different communication models, deployment methods and network density.

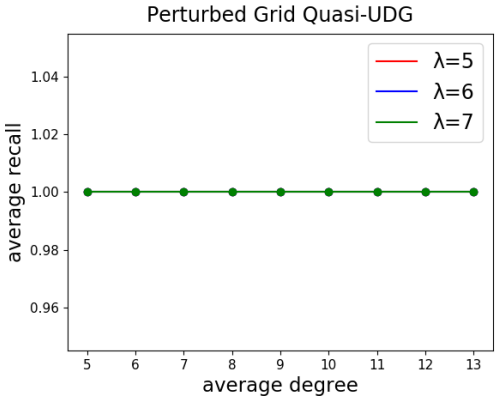
References

[1] Panos Papadimitratos and Zygmunt J. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Commnication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, pages 193–204, 2002.

[2] Kimaya Sanzgiri, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth M. Belding-Royer. A secure routing protocol for ad hoc networks. In *10th IEEE International Conference on Network Protocols, 2002. Proceedings.*, pages 78–87, 2002.



(a)



(b)

Fig. 6: Results on a network with 900 nodes, using Quasi-UDG communication model and perturbed grid deployment. We adjusted the radius of the sensors the achieve different network densities. (a) shows the average number of false positives, and (b) displays the average recall. Each point corresponds to the average of 30 simulations with the same average degree. We ran our algorithm with $\lambda = 5, 6, 7$, to demonstrate how this parameter influences the outcome.

[3] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1976–1986, 2003.

[4] Lei Zhang, Ligang Liu, Craig Gotsman, and Steven J. Gortler. An as-rigid-as-possible approach to sensor network localization. *ACM Transactions on Sensor Networks (TOSN)*, 6:35:1–35:21, 2010.

[5] Srdjan Čapkun, Levente Buttyán, and Jean-Pierre Hubaux. Sector: Secure tracking of node encounters in multi-hop wireless networks. In *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 21–32, 2003.

[6] Weichao Wang, Bharat Bhargava, Yi Lu, and Xiaoxin Wu. Defending against wormhole attacks in mobile ad hoc networks. *Wireless Communications and Mobile Computing*, 6:483–503, 2006.

[7] Lingxuan Hu and David Evans. Using directional antennas to prevent wormhole attacks. In *Proceedings of the Network and Distributed System Security Symposium Conference (NDSS)*, pages 241–245, 2004.

[8] Issa Khalil. Liteworp: A lightweight countermeasure for the wormhole attack in multihop wireless networks. In *Proceedings of the 2005 International Conference on Dependable Systems and Networks*, pages 612–621, 2005.

[9] Issa Khalil, Saurabh Bagchi, and Ness B. Shroff. Mobiworp: Mitigation of the wormhole attack in mobile multihop wireless networks. In *2006 Securecomm and Workshops*, pages 1–12, 2006.

[10] Radha Poovendran and Loukas Lazos. A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks. *Wireless Networks*, 13:27–59, 2007.

[11] Levente Buttyán, László Dóra, and István Vajda. Statistical wormhole detection in sensor networks. In *European Workshop on Security in Ad-hoc and Sensor Networks*, pages 128–141, 2005.

[12] Lijun Qian, Ning Song, and Xiangfang Li. Detection of wormhole attacks in multi-path routed wireless ad hoc networks: A statistical analysis approach. *Journal of Network and Computer Applications*, 30:308–330, 2007.

[13] Ritesh Maheshwari, Jie Gao, and Samir R. Das. Detecting wormhole attacks in wireless networks using connectivity information. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 107–115, 2007.

[14] Weichao Wang and Bharat Bhargava. Visualization of wormholes in sensor networks. In *Proceedings of the 3rd ACM Workshop on Wireless Security*, pages 51–60, 2004.

[15] Yi Shang, Wheeler Ruml, Ying Zhang, and Markus P. J. Fromherz. Localization from mere connectivity. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 201–212, 2003.

[16] Dezun Dong, Mo Li, Yunhao Liu, and Xiangke Liao. Wormcircle: Connectivity-based wormhole detection in wireless ad hoc and sensor networks. In *2009 15th International Conference on Parallel and Distributed Systems*, pages 72–79, 2009.

[17] Xiaomeng Ban, Rik Sarkar, and Jie Gao. Local connectivity tests to identify wormholes in wireless networks. In *Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 13:1–13:11, 2011.

[18] David Eppstein. Arboricity and bipartite subgraph listing algorithms. *Information processing letters*, 51:207–211, 1994.

[19] Xiaopei Lu, Dezun Dong, and Xiangke Liao. Mds-based wormhole detection using local topology in wireless sensor networks. *International Journal of Distributed Sensor Networks*, page 145702, 2012.