GAPs - A Geometric and Probabilistic Based Routing Scheme for Source Location Privacy in Wireless Sensor Networks

Aaron Ogle
Missouri State University
Springfield, MO

I. Introduction

Wireless Sensor Networks (WSNs) are used in many applications, some of which are sensitive applications, such as asset monitoring. With the growth of the Internet of Things (IoT), and the increased popularity of utilizing WSNs, it is critical that the networks achieve high source location privacy protection. Source location privacy (SLP) is defined in [1] as "the process of trying to minimize the traceability or observability of a source node by an adversary in a wireless sensor network" [1]. By source node, we mean the node that relays packets to the sink node in the network after it has detected movements made by an asset. The sink node in the network is the node that relays data to the outside world.

If an adversary is able to find the source node in a WSN then they will be able to locate the asset that the network is monitoring. Numerous routing schemes have been created in order to try and maximize the SLP protection in wireless sensor networks, a few of these schemes will be discussed below in the related work section.

In this work, we propose a new SLP based routing scheme that will rely on geometric and probabilistic methods. The proposed scheme will consist of partitioning the WSN into different tiers based off of a geometric technique. Once these tiers have been constructed, a route will be determined based off of randomly selecting a tier and randomly selecting a point in the tier for the source node to relay the packet to; we will denote this relay node as the selected path node. Once a path node is selected then the packet is routed to the path node utilizing a shortest path route. Once the path node obtains the packet it then forwards the packet to the sink node by utilizing a shortest path route.

To measure the efficiency of the algorithm, we implement an adversary in the program that will start at the sink node in the network and they will wait for packets to be sent to it so that they can perform a back tracing attack. There are a few different ways to measure the length of Safety Period, but in this work we utilize the definition provided in [1] where the Safety Period is defined as "the number of packets successfully delivered to the sink node before an adversary reaches the source node" [1]. The adversary model will be discussed in greater detail in Section III. We compare the Safety Period of the GAPs algorithm against the Shortest Path algorithm for different numbers of nodes in the network and find that the GAPs algorithm performs better than a Shortest Path algorithm when it comes to increasing the Safety Period for source location privacy in the network.

Along with comparing the Safety Period of the GAPs algorithm against the Shortest Path algorithm, we will also investigate how its energy consumption compares with the energy consumption of the SP algorithm.

The below work will be divided as follows. Section II will consist of related works and a literature review for the works. In Section III we will discuss the different models, such as adversary model and network model, that are used in this research and elaborate more on the exact problem we are solving. In Section IV we will discuss our proposed GAPs algorithm in detail. In Section V we will discuss our experimental results when comparing the GAPs algorithm to a Shortest Path algorithm. In Section VI we will discuss the conclusion of the work along with ideas for future work on this problem, and in the final section we will include the relevant references to the works mentioned throughout this research.

II. Related Work

Numerous routing schemes for SLP have been created which can be seen in [1] - [7] and [9].

The work in [1] proposes a path node offset routing algorithm. The algorithm first divides the sensors into two domains, which are classified as the near-sink region, and the region that is away from the sink [1]. The algorithm computes the offset angle of the nodes which begins by constructing two imaginary lines, one line connects the source node to the sink node, and the other line is the line connecting the source node to the X-axis [1]. The distance between the source node and sink node is computed using the euclidean distance metric, and the distance between the source node and the X-axis is also computed using the euclidean distance metric; once these distances are calculated then the offset angle can be found from them [1]. After computing the offset angle, the source node determines what region it is in, determines a set of candidate path nodes to relay the packet to, and then computes an arbitrary factor [1].

Based off of the offset angles and the arbitrary factor, a path node is selected for the source node to route the packet to. The limitation in this design is that it leads to higher energy consumption and delivery latency when compared to other algorithms, such as the Phantom Single-Path algorithm.

In [2] the authors propose a routing scheme that utilizes diversion nodes and mediate nodes, the difference between the diversion and mediate nodes lies in which region they are located in; these nodes are implemented in order to make it difficult for an adversary to trace packets back to the source node [2].

The algorithm works by dividing the nodes in the network into two regions, one of which is the ”hotspot region”, the other is the ”non-hotspot” region [2]. The hotspot region is the region which is nearest to the sink node, and the non hotspot region lies outside of this one [2]. A value is determined based off of a radius from the sink node to the outer edge of the diversion node region, denote this value as X, the difference between this region and the hotspot region is where the diversion nodes are located [2]. Another value is constructed based off of the radius from the sink to the outer layer of the mediate node region and the difference between this region and the value denoted X above is where the mediate nodes are located at [2]. The algorithm constructs routes to the sink node from the source node based off of these regions, neighbor nodes, random numbers, and different thresholds [2].

The work in [3] introduces a scheme that uses proxy nodes in order to confuse adversaries. The algorithm is deployed in two phases where phase 1 consists of having packets be randomly forwarded by the source node to a random proxy node that is located in proxy regions which have been predefined, the design of where the proxy nodes are located at is constructed in order to make it seem like the source node is sending packets to the sink node from different directions [3]. In phase 2 of the algorithm, the proxy node will randomly send a packet to the sink node utilizing a random walk routing path; the scheme also uses a random factor and a threshold value in order to guarantee that routing paths for packets will be different, with this highly random scheme it will make it difficult for an adversary to try and predict where the next route of a packet will be [3].

In [4] along with constructing a new routing scheme the authors also propose an adversary model that is more intelligent than ones that are considered in other works [4]. In the author’s model, the adversary is intelligent enough to be able to find the type of packet by checking the header of each packet; using this information the adversary then uses a Hidden Markov Model in order to try and find what state the source is in for a given time period [4].

Due to implementing a more advanced adversary, the authors utilize phantom nodes and fake source nodes in order to try and achieve better source location privacy protection [4]. Phantom nodes are nodes which are placed near the source node which mimic the behavior of the source node; fake source nodes also mimic the function of the source node but these nodes are located around the sink region [4]. To highlight how the algorithm works, the algorithm first chooses phantom nodes around the source node, the algorithm then has each node calculate a weight value in order to determine next hop candidate nodes for the route; after these steps, fake sources are created around the sink in order to send fake packets with the hope that these fake packets confuse the adversary [4].

The works discussed so far have considered models which consist of one sink node, but the work done in [5] proposes a route that is based on multiple sink nodes. The steps involved in their routing algorithm consist of first selecting a phantom node randomly, and then the area around this node is constructed by a technique called flooding, where flooding is a method where a node routes packets to its neighboring nodes [5]. The next step of the algorithm consists of slicing the real packets into different shares where each of these shares is transmitted to a selected sink node; after these shares are constructed the algorithm then has the sink node send a fake packet in order to form a loop after it receives one of the shares [5]. The reasoning behind slicing the packets into different shares is so that the the shares can be routed through multiple dynamic paths; according to the authors this will increase the backtracking difficult for the adversary that is in the domain [5].

The construction of the dynamic multipath routing scheme is based off of nodes determining their destination sink node based off of the quadrant that it is in, the node will then calculate an angle between itself and the destination sink node, the forwarding node also finds the angle between itself and its neighbor nodes [5]. After this process is completed, based off of the angle, and a few other conditions based off of the location of the nodes neighbor, a path based off of different next hop candidates is created [5].

In [6] the authors also work on a scheme that involves the use of multiple sink nodes. The authors state that they use multiple sink nodes in order to create many routing paths [6]. For their scheme, the authors propose an algorithm that randomly selects a sink from multiple sink nodes during the first phase, the sink selection process is useful due to the aspect of changing the routes so that it makes it difficult for an adversary to back trace in the network [6]. After a sink node is selected, routing paths are changed dynamically, and packets are sent to nodes that are in different regions in order to alleviate the problem of having certain hotspots in the network, these nodes are referred to as intermediate nodes [6]. By hotspots it appears that the authors are referencing an area of the sensor network that might be visited more than others and they want to avoid this. The final stage of the scheme uses fake hotspots and fake branches to make the traffic in the network more complex, where fake branches are paths in the network which have dummy packets in them which are intended to confuse the adversary [6]. With their scheme the authors find that having their strategy makes it difficult for adversaries in the WSN to trace packets due to the scheme ensuring random and safe routing [6].

Instead of just focusing on one asset that is being monitored in a wireless sensor network, the authors in [7] propose a routing scheme for the scenario where there are multiple source nodes, or in other words, multiple assets that are being monitored. In their work the authors propose two new phantom routing algorithms designed to protect the source location privacy of multiple assets that are being monitored in a WSN [7]. The first algorithm that the authors implement consists of first having every node divide the neighboring set of nodes into three groups named farther neighbors, equal neighbors, and closer neighbors which is based on the distance between the nodes [7]. These groupings of nodes are given set names that are closer-set, farther-set, and equal-set; the nodes that are in the equal-set and farther-set are then combined so that a set called backward-set can be constructed [7]. After the sets are created, and once a source node detects the asset, then the source node will randomly choose a neighbor that is in the backward-set and will send the packet to that node; utilizing a time metric, the packet is sent to neighboring nodes until a hop count in the process equals zero, once this is done the node that is holding the packet at that time becomes a phantom node [7]. After this is done, the phantom node will send the packet to the sink node, or base station, as it is referred to in this paper, using a greedy routing method [7].

Along with this algorithm, the authors also propose another one that consists of three phases. The first phase of the scheme consists of having the source node randomly send a packet to one of its neighbors, then that neighbor randomly routes to another neighbor; along with this random routing there is also a time metric which is decreased during the routing process and after the route goes through a certain number of hops the next phase of the algorithm starts [7]. The node that receives the packet last in the method described above becomes a phantom node, and the phantom node then invokes what the authors refer to as the L-Walk phase [7]. The L-Walk phase consists of having the packet move in vertical and horizontal directions, where the choice of sending the packet in a vertical or horizontal manner are determined based off of different criteria; during this process a time metric is also utilized and is decremented through the process and once the time count metric reaches zero then the packet is sent to the sink node via a shortest path routing phase [7].

In [9] the authors propose a routing method that consists of constructing an anonymity cloud based source location privacy protection scheme that protects against an enhanced hotspot locating attack [9]. The author's proposed method involves splitting data packet messages based off of utilizing congruence equations [9]. Once the messages are split into different "shares" the scheme then constructs an anonymity cloud based off of these shares in order to hide the location of the asset or source node in the network [9]. While the cloud is being constructed in the network the scheme also has fake packets generated and spread through the network in order to protect the real shares of the message from being located by the adversary [9]. Through various conditions the real shares and fake shares are routed in the network and then the real shares are sent to the sink node based off of a hop count being decremented; once the sink node receives the real shares it is able to reconstruct the message once it has a certain number of the real shares of the original message [9]. From the author's results in their paper it looks like this method performs very well when it comes to minimizing the source node detection probability [9].

III. Models and Problem Statement

In this section we will discuss the structure of the WSNs that are used in this work, the adversary's behavior, and the asset's behavior.

For this project, we will utilize networks that consist of a varying number of sensors. The first network model will consist of 169 sensors, the second will consist of 225 sensors, the third 289 sensors, and the fourth model will use 361 sensors. We utilize networks of varying sizes so that we can perform experiments on how the Safety Period changes for the algorithms based off of the number of nodes that are in the network.

An important thing to note about the network model that is implemented in this research is that it is a grid type network instead of a mesh type network. From this we mean that the nodes in the network can only relay packets to neighbor nodes that are above, below, to the left, and to the right of the node instead of being able to transmit packets to diagonal neighbors. This network structure leads to shortest path routes that do not take diagonal paths due to nodes not being able to pass packets to their diagonal neighbors.

For this project, the network models will consist of sensors which can be thought of as points in the two-dimensional plane. These points (sensors) are evenly distributed in the plane. For some tests, the sink node will be the node that is in the center of the network, and in other tests it will be the node that is located in the bottom left corner of the network.

Our points are numbered starting with (0,0) in the bottom left corner going up to (12,12) in the top right corner for the network model that consists of 169 sensors. In order to test the efficiency of the proposed algorithm we will place the sink node in different locations of the network to see how this changes the results of the number of packets that are successfully sent before the adversary is able to locate the source node. The location of the sink node will first be in the middle of the network, at point (6,6), tests will also be ran on the network when the sink node is located in the bottom left corner of the network at point (0,0). The point of changing the location of the sink node is to see if this effects the efficiency of the proposed algorithm at all. Figure 1 on the following page shows our 169 sensor network model in the 2D plane and the image below that shows our 361 sensor model in the 2D plane.

For our network model that consists of 361 sensors, the sink node will be in the middle of the network at point (9,9), and then for an additional test it will be located at point (0,0); the same logic applies to finding the sink node for our other two network models.

The adversary is similar to the one proposed in [1]. In this model the adversary will start at the sink node and will listen for packets that arrive to it. The adversary will only be able to travel to nodes that are directly next to it. Since we are using points in a 2D plane, the adversary will have a total of eight

Fig. 1. Network Sensors

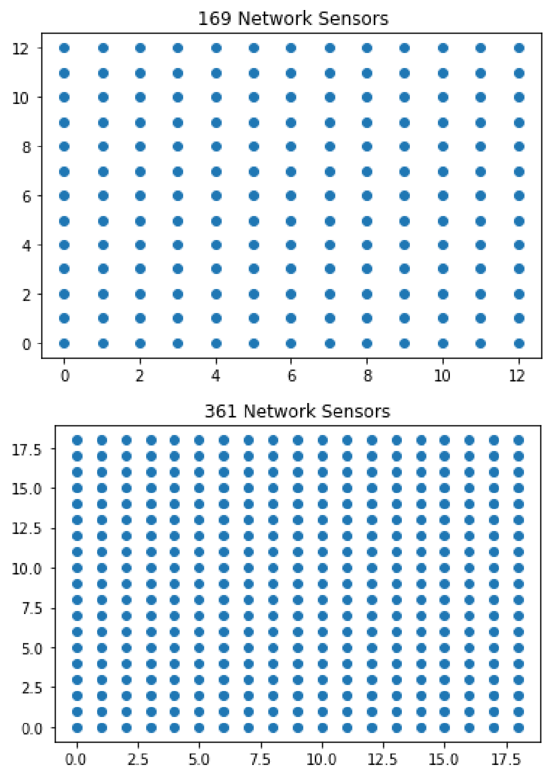
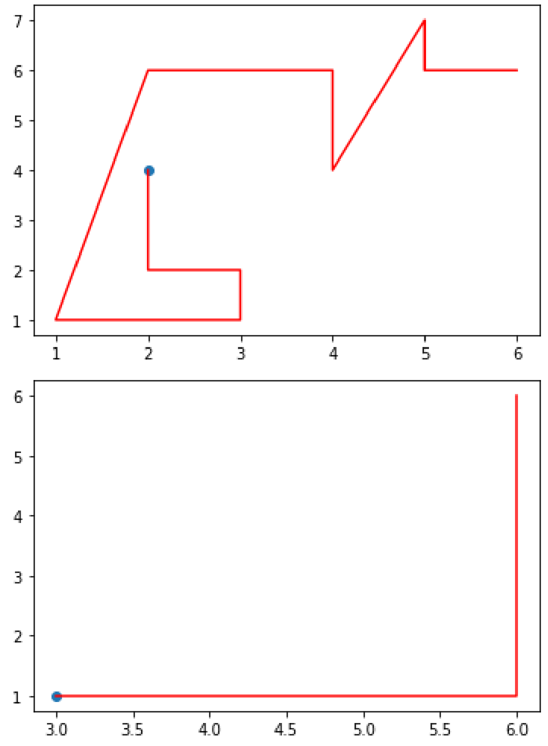


Fig. 2. Adversary Paths - GAPs and SP



neighbor nodes around it at any given time. For example, if the adversary begins at point (6,6), the sink node for our 169 sensor model, then the adversaries beginning neighbor nodes are the following set of points - [(6, 7), (6, 5), (5, 6), (7, 6), (7, 7), (5, 7), (7, 5), (5, 5)]. If a packet is routed through any of these points then the adversary will move to that point and will calculate its new set of neighbor nodes. Every time that the adversary moves in the network they will calculate what nodes are its neighbor nodes and then they will wait for a packet to be routed through one of them and if a packet is routed through one of them then the adversary moves to that node. The adversary repeats this process until they are able to reach the location of the source node. Figure 2 shows paths taken by the adversary in our GAPs implementation and also the Shortest Path Algorithm implementation.

For the figures, we utilized our 169 sensor model and removed all of the nodes except for the source node in order to highlight the path taken by the adversary. In the first image, we can see the adversary begins at the sink node, point (6,6), and after some time they are finally able to locate the source which is at point (2,4). In the second image, we can see that adversary begins at the point (6,6) and they are able to take a more linear path to the source node which is at the point (3,1). These images are used in order to highlight the fact that the adversary has a much more difficult time tracking down the source node in the GAPs algorithm implementation than they do with a shortest path routing algorithm. Direct results of the simulations will be discussed in more detail in the results section, but these images are just used in order to highlight the adversaries behavior in our network models.

In the below section we highlight the pseudocode used for our adversary model.

```
$\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#$
$Adversary\_Model$
$\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#$
$Input \leftarrow adversary$
$Input \leftarrow adversary\_route\_list$
$Input \leftarrow packet\_path$
$Compute\ Adversaries\ Neighbor\ Nodes$
**if** $NeighborNode\ in\ packet\_path$ **then**
    $adversary = NeighborNode$
    $adversary\_route\_list.append(adversary)$
**end if**
$Output \leftarrow adversary,\ adversary\_route\_list$
```

Now that we have looked into the adversary model in our network we can turn our attention to the model of our asset in the network.

For this model, we assume a lazy asset, where after the source node picks up on their location the asset will no longer move. This will cause the source node that is originally generated to be the only source node throughout the simulation; however, multiple simulations will be ran so that the source node will be changed through each simulation.

Now that our models for this study have been defined we will now directly define the problem that we are trying to address.

As mentioned above, source location privacy (SLP) is defined in [1] as ”the process of trying to minimize the traceability or observability of a source node by an adversary in a wireless sensor network” [1]. For our study, we are defining a Safety Period to be the total number of packets that are successfully sent in the WSN before the adversary is able to locate the asset, or source node. In our study, we want to maximize the value of this Safety Period. We want to maximize the number of packets that are successfully sent to the sink node before the adversary is able to locate the asset in the network. We aim to implement the GAPs algorithm so that it can maximize the Safety Period in a WSN and we will compare the results of our algorithms Safety Period value with the Shortest Path algorithms Safety Period value.

IV. Proposed Geometric and Probabilistic Based Routing Algorithm

Before discussing the steps of the GAPs algorithm in full detail, we will first highlight the process of partitioning our network with images of the convex hull partitioning process. As we will discuss later, the GAPs algorithm partitions a network into different tiers by finding the convex hull of different sets of points in the network. The following images show the partitioning process for the network model that contains 361 sensors.

Fig. 3. First Iteration of Convex Hull Process with 361 Sensor Network

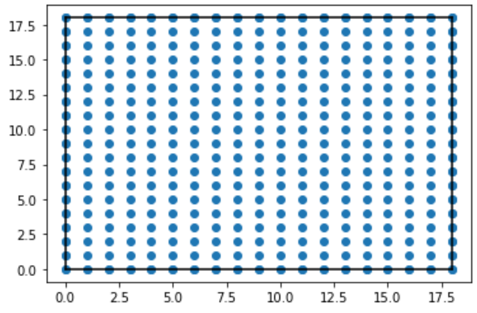
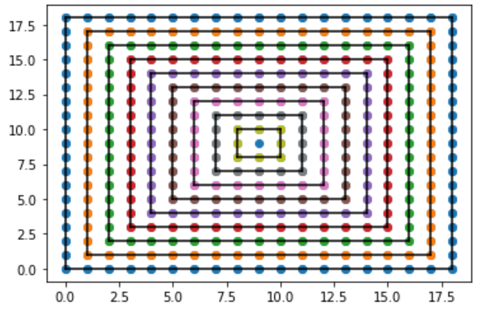


Fig. 4. Final Iteration of Convex Hull Process with 361 Sensor Network



In the figures we have a network that consists of 361 sensors. The GAPs algorithm partitions the network into 9 different tiers. Now that the network has been partitioned the GAPs algorithm will randomly select one of these 9 tiers, and then once a tier is selected the algorithm will randomly select a point in that tier. Once the point is selected we call this point a ”path node”. Once the path node is chosen, the source node forwards the packet to this path node using a shortest path route. Once the path node obtains the packet it forwards the packet to the sink node using a shortest path route. Now that we have looked into a visualization of the network partitioning process and a brief example of the GAPs algorithm we will now look into the algorithm in full detail.

The Geometric and Probabilistic based Routing Scheme (GAPs) will consist of the following steps:

1) The network will be divided into tiers based off of a convex hull implementation process where the convex hull of a planar set is defined as the minimum area convex polygon containing the planar set [8]. Given the initial set of points the convex hull of those points will be computed and the points that lie in this convex hull will be added to a list; after this is done these points will be removed from the initial set and the convex hull of the remaining points will be computed. This process will be repeated until there is only one point left in the original input set.
2) After the different tiers have been constructed, and the source node picks up on the assets movement, then the source node will randomly select one of the tiers, after randomly selecting one of the tiers the source node will randomly select one of the points that is contained within this tier. The point that is selected will be referred to as the path node.
3) Once the path node is selected the source node will send the packet to the path node through the shortest routing path. After the path node obtains the packet then the path node will forward the packet to the sink node through the shortest routing path.

To help understand the partitioning process of the algorithm, let us look into the situation when we are partitioning the network that consists of 169 nodes.

1) First, the algorithm accepts the set of points, or sensors, as input.
2) Next, we compute the convex hull on the total set of points given.
3) Since our set of points will be in the range of [0,0] to [12,12], we know that the vertices which are contained in the first iteration of the convex hull will contain the coordinate values of 0 and 12, then the next iteration will contain the coordinate values of 1 and 11, and so on and so forth until there are no more nodes to check. Because of this geometric structure of our points we can create a points permutations list which essentially sorts our sensor nodes based off of the order of which they lie in the computed convex hull.
4) With the above geometric property we find the values of the x coordinates that are contained in the geometric hulls vertices.
5) We then iterate through all of the sensors in our sensor

list and check and see if their x coordinate values are the same as the ones that are contained in the geometric hulls vertices list.
6) If the vertices match then we add the point to our points permutations list and remove them from our original sensor list.
7) By removing the points from the original list of sensors we are able to perform the process again so that we can compute the different convex hulls of the different layers of the network.

The following pseudocode highlights the process of the partitioning process.

```
$################################$
$ConvexHull\_Partition$
$################################$
$Input \leftarrow sensors$
$point\_permutation \leftarrow []$
$num\_of\_sensors \leftarrow []$
$count \leftarrow 0$
while true do
    $num\_of\_sensors.append(len(sensors))$
    $count += 1$
    $Compute\ Convex\ Hull$
    $Compute\ Vertices\ in\ Convex\ Hull$
    for sensor in sensors do
        if Convex Hull Vertices in sensor then
            $Add\ point\ to\ points\ permutation\ list$
        end if
    end for
    for sensor in points permutation list do
        if sensor in original sensor list then
            $Remove\ point\ from\ original\ sensors\ list$
        end if
    end for
    if Only one point remains in original sensors list then
        $Break\ out\ of\ while\ loop$
    end if
end while
```

Once we have the ”points permutation” list, the total count of tier layers, and the number of sensors that belong to each tier, we can then take this data and construct a matrix that represents the different tiers of the partitioned network.

Now that we have a better idea of how the partitioning process works, let us take a look at the pseudocode for the implementation of the GAPs algorithm.

```
$################################$
$GAPs\_Algorithm$
$################################$
$Input \leftarrow partitioned\_tier\_matrix$
$Input \leftarrow source\_node$
$Set\ Sink\ Node\ Location$
$Set\ Adversary\ Start\ Location$
$adversary\_route\_list \leftarrow []$
$Add\ Adversary\ Start\ to\ Route\ List$
$count \leftarrow 0$
while true do
    $count += 1$
    $Select\ Random\ Tier$
    $Select\ Random\ Point\ in\ Tier$
    $path\_node \leftarrow Random\ Point$
    $Compute\ SP\ to\ PN$
    $Compute\ SP\ from\ PN\ to\ SN$
    $Send\ Adversary\ Path\ Information$
    $Call\ Adversary\ Function$
    if Source Node is in Adversaries Route List then
        $Break\ Out\ of\ While\ Loop$
    end if
end while
$Output \leftarrow count$
```

In the provided pseudocode, by SP we mean shortest path, and by PN we mean path node. The function returns the value of ”count” which will represent the total number of packets that were successfully sent to the sink node before the adversary was able to locate the source node.

V. Experimental Evaluation

For the performance analysis of our proposed GAPs algorithm we compare the Safety Period for its performance to the Safety Period performance of a shortest path routing algorithm.

In order to investigate the performance of our proposed GAPs algorithm we will utilize networks that contain different numbers of sensor nodes. We will investigate networks that contain 169 sensors, 225 sensors, 289 sensors, and 361 sensors. Along with measuring the performance of the GAPs algorithm and Shortest Path algorithm in networks of varying sizes we will also investigate the performance of the algorithm when the location of the sink node changes. We will first let the sink node be placed in the middle of the network and then we will re locate the sink node to the bottom left corner of the network.

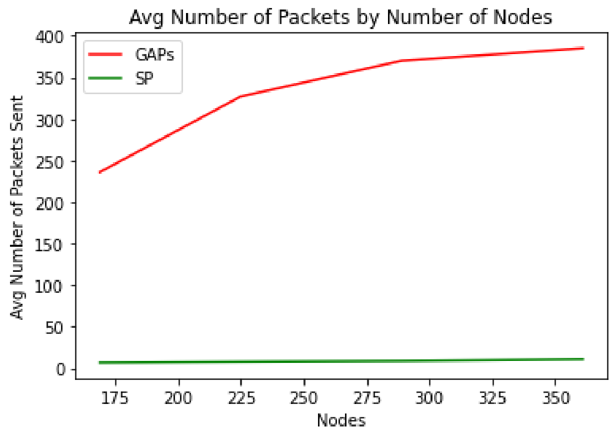
During our experiments we will randomly generate the location of the source node and then run the GAPs algorithm and Shortest Path algorithm in order to obtain the number of packets that are successfully sent from the source node to the sink node before the adversary is able to locate the location of the source node. In one of our tests we will also pick a stationary location for the source node so that we can check the performance of the algorithms when the source node is no longer moving around randomly for each simulation.

For our simulations we will run a total number of 100 tests for each scenario described above. We will then calculate the average number of packets successfully sent during these 100 simulations to utilize for our results.

For our first performance analysis test, we utilize the following network structures. In our first simulations we utilize a network of 169 nodes, with the sink node located in the center of the network at point (6,6); this is also the location of where our adversary starts at. With this network structure we run 100 simulations where we randomly generate a location for a source node, then call the GAPs algorithm, and SP algorithm to see how many packages are successfully sent during each simulation. After this is done, we then calculate the average number of packages sent during the simulation by both methods. We repeat this process for networks consisting of 225 nodes, 289 nodes, and 361 nodes. For each of these networks we place the sink node in the middle of the network. In the below table and chart we show the results obtained for our first test. The table and chart shows the average number of packets sent by both the GAPs algorithm and the SP algorithm before the adversary is able to locate the source node along with the corresponding nodes that were utilized in the network.

Avg Number of Packets Sent by Number of Nodes

Nodes	GAPs	SP
169	236	7
225	327	8
289	370	9
361	385	11



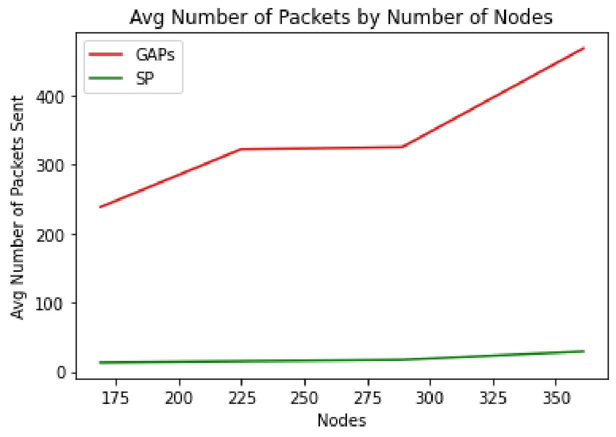
From the results we can see that the GAPs algorithm was able to transmit far more packets than the shortest path algorithm. From the results we can also see that the shortest path algorithm's efficiency does not grow very much when the number of nodes in the network increases.

This makes sense intuitively due to the number of successful packets that can be sent by the shortest path algorithm before the adversary can locate the asset essentially just becomes the total length of the path between the source node and the sink node.

For the second performance analysis test, we keep everything the same as it was in the first test but instead of having the sink node located in the center of the network we place the node in the bottom left corner of the network. After making this modification the below results were obtained:

Avg Number of Packets Sent by Number of Nodes

Nodes	GAPs	SP
169	238	13
225	322	15
289	325	17
361	468	29



From the results we can see that the shortest path method is able to transmit more packets when the sink node is located in the bottom left corner of the network than when it is located in the middle of the network; this again makes sense as the path length for the packets will be longer for the scenario when the sink is located in the bottom left corner of the network as opposed to when it is located in the center of the network.

Interestingly, the GAPs algorithm does not have a high increase in the number of packets sent when the number of nodes in the network changes from 223 to 289; a reason for this could be the probabilistic structure of the algorithm along with where the source node appears in the network; depending on what tiers and nodes are selected during the routing process could affect how easily it is for the adversary to back trace the packets and locate the source node.

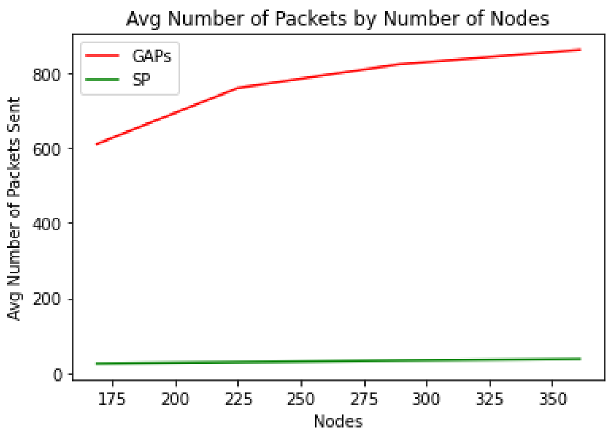
Even with the change of sink node location the GAPs algorithm was still able to transmit far more packets than the shortest path algorithm.

For the third performance analysis test, we again keep everything the same as in the above tests but we keep the sink node in the bottom left corner of the network and instead of having the source node be selected randomly we instead give it a fixed value. For this test, we will assume that the source node is located in the top right corner of the network, which will be as far away from the sink node as is possible in the network.

After making this modification the below results are obtained:

Avg Number of Packets Sent by Number of Nodes

Nodes	GAPs	SP
169	610	25
225	759	29
289	822	33
361	861	37



From the results we can see that the GAPs algorithm and the Shortest Path algorithm are both able to send more packets on average when the source node is located as far away as possible from the sink node in the network. This makes sense as the distance between the asset and the adversary would be the greatest that it can be in this scenario.

We again see a linear increase for the shortest path algorithm in this scenario which again makes sense as the total number of packets that the algorithm can send will essentially just become the length of the total path from the sink node to the source node.

It is again important to highlight here that our network is a grid structure which means that the nodes in the network are unable to transmit packets to their diagonal neighbors.

Even with the source node being located as far away as possible from the sink node in the network the GAPs algorithm is still able to transmit far more packets than the shortest path algorithm which shows that the GAPs algorithm can achieve much higher source location privacy protection than a simple shortest path algorithm can.

Now that we have compared the algorithms by their Safety Period performance, let us look into how the algorithms compare when it comes to energy consumption.

In order to calculate the energy consumption of our model, we utilize the same parameters as those that were used in [1]. To simplify things, we make some extra assumptions about our model. First, we assume that all of the nodes in the network are approximately 20 meters away from all of their neighbor nodes. Secondly, due to our first assumption, we do not utilize a distance threshold like the authors did in [1].

This causes the energy transmission equation to always be

$$E_{trans} = lE_{loss} + lE_{fs}d^2$$

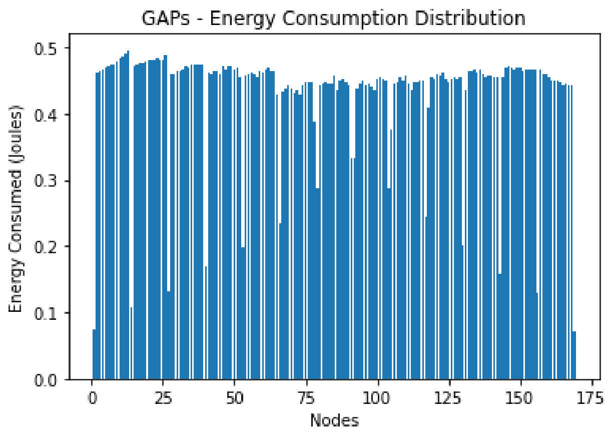
and the energy reception to always be

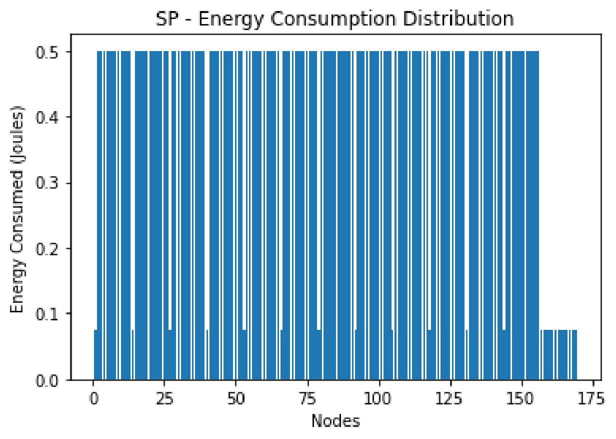
$$E_{rec} = lE_{loss}$$

by assuming that d = 20 meters, and by substituting in the values provided in [1], our energy reception becomes 5.12e-5 joules, and our energy transmission becomes 5.5296e-5 joules. This means that any time a node receives a packet, it burns through 5.12e-5 joules of energy, and anytime that it transmits a packet it burns through 5.5296e-5 joules of energy.

With this, the total energy consumed by a node will be the addition of the energy transmission and the energy reception which is 1.06496e-4 joules. For testing the difference in energy consumption between the GAPs and SP algorithm, we want to investigate what the distribution of energy consumption is amongst the sensor nodes. To do this, similar to the work done in [1], we assign each node in the network with an initial energy value of 0.5 joules. Each time that a node has a packet routed through it, we subtract 1.06496e-4 joules from that value. By doing this, we can get an idea of how distributed the energy consumption is throughout the network.

For our test, we utilize the network of 169 sensors. We place the sink node in the bottom left corner of the network at point (0,0) and we place our source node at point (12,12). We then have each scheme route 3500 packets in the network and we keep track of the energy that is consumed. In the below plots, we show the energy consumed by the nodes in the network by using the GAPs algorithm, and the energy consumed by the nodes in the network by using the SP algorithm. With these images we can see that there is more of a balanced energy distribution by using the GAPs algorithm than there is by using the SP algorithm. The SP algorithm uses the same nodes for its routes which will cause their energy to deplete faster.





From the results we can see that the GAPs algorithm utilizes far more nodes for routing in the WSN than the SP algorithm does. If we look at the results for the SP algorithm, we can see that the same nodes look like they are being used each time, which makes sense given the goal of the shortest path scheme. The drawback of the shortest path scheme lies in the fact that it uses the same nodes every time for routing the packets to the sink node; by doing this, the node's energy will be depleted at a faster rate due to them constantly being used to route packets to the sink node.

VI. Conclusion and Future Work

In this research work we have discussed what source location privacy protection is and have done a review of current research work that is being done on creating source location privacy routing schemes that are implemented in order to try and maximize the source location privacy protection in the network. After reviewing the work that is currently being done on this problem, we have proposed a new source location privacy protection algorithm and have gone over its implementation in detail.

After discussing the algorithm in detail we have shown that the algorithm performs far better when it comes to source location privacy protection than a simple shortest path algorithm does. In each of the three different experimental tests performed the GAPs algorithm is able to transmit far more packets to the sink node before the adversary reaches the source node than the shortest path algorithm can.

Along with comparing the Safety Period of the GAPs algorithm with the SP algorithm, we have also investigated energy consumption by the different schemes, and have found that the GAPs algorithm's energy consumption is more distributed amongst the nodes in the network than the SP algorithm's is.

For future work we will compare the GAPs algorithm to other algorithms that are currently utilized for SLP in WSNs such as the various algorithms that are discussed in the literature review above. Specifically, we would first like to compare the GAPs algorithm's performance to the performance of the Path Node Offset Angle Routing Algorithm that is discussed in [1] as this algorithm utilizes various geometric techniques as well. In future work we also hope to measure the algorithms performance when it comes to different metrics such as packet delivery latency.

For future source location privacy protection routing schemes we would like to look into different machine learning algorithms such as reinforcement learning and see if any techniques from that area could be utilized in order to help with source location privacy protection in wireless sensor networks.

References

[1] L. C. Mutalemwa and S. Shin, ”Regulating the Packet Transmission Cost of Source Location Privacy Routing Schemes in Event Monitoring Wireless Networks,” in IEEE Access, vol. 7, pp. 140169-140181, 2019, doi: 10.1109/ACCESS.2019.2943710.

[2] L. C. Mutalemwa and S. Shin, ”Strategic location-based random routing for source location privacy in wireless sensor networks,” Sensors, vol. 18, no. 7, p. 2291, 2018.

[3] L. C. Mutalemwa and S. Shin, ”Achieving source location privacy protec- tion in monitoring wireless sensor networks through proxy node routing,” Sensors, vol. 19, no. 5, p. 1037, 2019

[4] H. Wang, G. Han, W. Zhang, M. Guizani and S. Chan, ”A Probabilistic Source Location Privacy Protection Scheme in Wireless Sensor Networks,” in IEEE Transactions on Vehicular Technology, vol. 68, no. 6, pp. 5917-5927, June 2019, doi: 10.1109/TVT.2019.2909505.

[5] G. Han, H. Wang, X. Miao, L. Liu, J. Jiang and Y. Peng, ”A Dynamic Multipath Scheme for Protecting Source-Location Privacy Using Multiple Sinks in WSNs Intended for IIoT,” in IEEE Transactions on Industrial Informatics, vol. 16, no. 8, pp. 5527-5538, Aug. 2020, doi: 10.1109/TII.2019.2953937.

[6] G. Han, X. Miao, H. Wang, M. Guizani and W. Zhang, ”CPSLP: A Cloud-Based Scheme for Protecting Source Location Privacy in Wireless Sensor Networks Using Multi-Sinks,” in IEEE Transactions on Vehicular Technology, vol. 68, no. 3, pp. 2739-2750, March 2019, doi: 10.1109/TVT.2019.2891127.

[7] M. R, T. Koduru and R. Datta, ”Protecting Source Location Privacy in IoT Enabled Wireless Sensor Networks: the Case of Multiple Assets,” in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2021.3126171.

[8] Eddy, William F. ”A new convex hull algorithm for planar sets.” ACM Transactions on Mathematical Software (TOMS) 3.4 (1977): 398-403.

[9] N. Wang, J. Fu, J. Li and B. K. Bhargava, ”Source-Location Privacy Protection Based on Anonymity Cloud in Wireless Sensor Networks,” in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 100-114, 2020, doi: 10.1109/TIFS.2019.2919388.