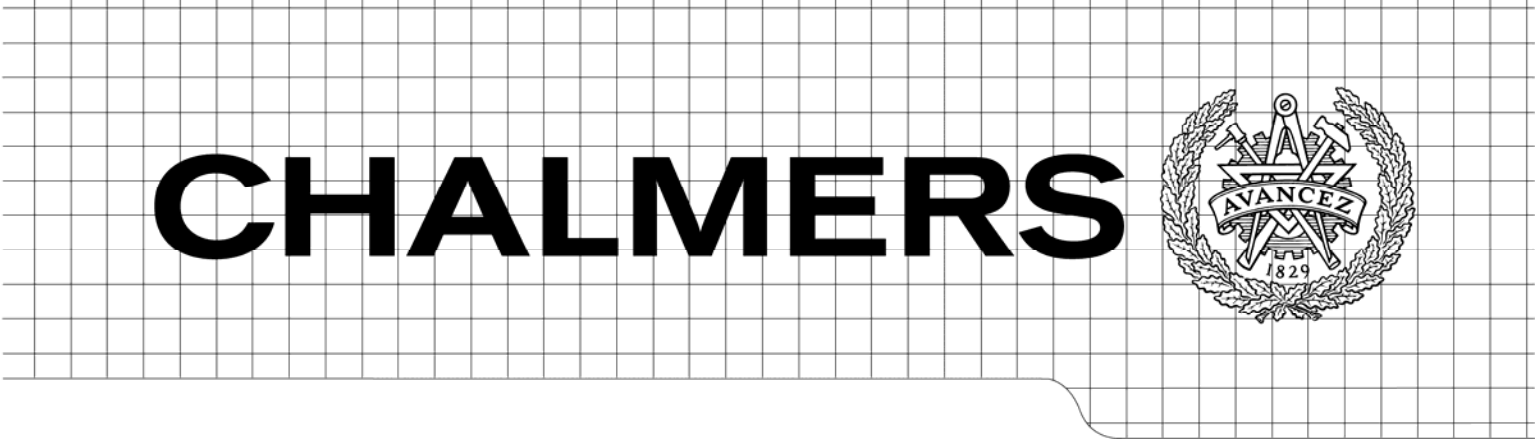

Secure Geographic Routing in Wireless Sensor Networks

Master of Science Thesis in the Programme Network and Distributed Systems

FARINAZ GHASEMI

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, January 2013

Secure Geographic Routing in Wireless Sensor Networks

FARINAZ GHASEMI

Examiner: Philippas Tsigas

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden January 2013

Abstract

Many routing protocols have been designed and implemented for the ongoing and emerging technology of Wireless Sensor Networks. The main objective of these protocols is to overcome the limitations of sensor nodes and to prolong the network lifetime. Geographic Routing, which performs routing based on the geographic location of sensors, is proposed for large scale and highly dynamic sensor networks. Two of the best known of these protocols are GPSR (Greedy Perimeter Stateless Routing) and GEAR (Geographic energy Aware Routing). In the former, the protocol makes routing decisions using the positions of the nodes and the destination of the packet. While in the later in addition to the distance and location metrics, the energy level of the nodes is considered.

Since security is not considered in the design of these two protocols, they are both vulnerable to various types of attacks such as Sybil attack, Sinkhole attack, Selective forwarding and many others.

The current research is aimed to achieve four main goals. The first, analyzing the functionality of the two protocols. The second, putting the protocols under similar attacks in order to examine the impact. The third, detecting attacks through statistical analysis of network behavior. Finally the fourth, securing the protocols through mitigation techniques such as multipath routing and reputation/trust-based systems.

Keywords: Wireless Sensor Networks, Geographic Routing, Secure routing, GPSR, GEAR, Multipath Routing, Reputation System

Acknowledgements

I would like to thank:

My examiner "*Philippas Tsigas",* Thank you for accepting me as one of your students. For trusting me and giving me the opportunity to conduct this challenging and interesting research. It was a completely new concept to me and brought out priceless theoretical knowledge and practical experiences. I also thank you for the extremely worthwhile and elaborate Distributed Systems Advanced Course, which helped me to learn the materials that I was seeking to learn for a long time. Thank you for your invaluable guidance and the time you took despite your hectic and busy schedule.

My supervisor *"Andreas Larsson",* Thank you for all the discussions, meetings and being an example to me of a real professional programmer capable of developing such well performed, effective and robust codes. I am very appreciative to have had the chance to work with you.

My supervisor *"Olaf Landsiedel"*, I would like to thank you professionally and humanly. You not only enabled me to stay on the right path, but also helped me to converge my scattered and incoherent findings to structure them in harmony and consistency and bringing them to a designated end. I thank you for being so knowledgeable, enthusiastic, reassuring, committed and supportive.

"Marina Papatriantafilou", My heartfelt thanks to you for your invaluable help and support.

"Afshan Samani", Thank you for pre-paving my way and making things easier.

"Farid" and *"Mahdi"*, my late beloved brothers, my childhood buddies, my huckleberry friends, my confidants, my best companions ever. Now it is more than a year that you have left me, Maman and Baba behind. Thank you for all the love and care, laughs and tears, dreams and adventures, fights and arguments, crazy plans and ecstatic chases and for all the fantastic moments you shared with me along the road. I am so blessed to have had you two as brothers in my life. I thank you for all you taught me in your lives, and for redefining my every value with your premature and without warning end. Time changes everything, however, one truth always stays the same, you will be a part of me forevermore and I will always love you until some day we meet again.

Maman and Baba, I can never thank you enough for all you have been to me. Thank you for being my drive for going on and for not giving up. I wonder how you could be my pillar of strength from miles away, while you yourselves were bearing the most unbearable sorrow that any parent could endure. There is no way I could ever repay such a great devotion. I am forever indebted to you and I love you the way words cannot explain.

My little niece "*Miriam",* although just now, you have no idea why I am thanking you, you will know someday that you are keeping the light on inside of my heart and every single day, you are giving me a new reason to move on. I love you my little angel, thank you!

My dear sister-in-law "*Ashi",* Thank you for watching over the family on behalf of me over the past year and for holding the family chain connected.

All my precious friends in Iran and Sweden, in particular *"Hengameh Samifar*" "*Fatemeh Ayatollahi"* and *"Sareh Talebi"*, Thank you for being there for me in all the moments of joy and sorrow and being true friends in the most extensive meaning of friendship. I love you all!

Finally yet most importantly, my God, my shield, my shelter, my all. My deepest love and praise to you for your never ending support through all the moments of life. Thank you for all rises and falls, challenges, obstacles and tests you have put in my way to make me who you wanted me to be. Yet, thank you for tangibly proving me time after time that you are still in the business of miracles and protecting your people. I love you old timer, thank you!

Contents

List of Figures

List of Tables

List of Equations

1. Introduction

1.1 Motivation

A Wireless Sensor Network (WSN) is a network composed of large number of small, low power, low-cost devices called sensors. The main task of WSNs is to monitor and report environmental conditions. One of the main characteristics of WSNs is that all the sensors belong to a group and work toward a common goal. An individual senor node has little value of its own unless it works in cooperation with other sensors in a distributed fashion. They exchange messages frequently according to the application demands and report the information to a single or to multiple sinks.
In computer networks, the process of forwarding a message from source to destination through a series of intermediate relays nodes is called routing. Compared to wired networks, the topology of WSNs changes more frequently. Therefore, more update messages are sent and cause network congestion and enormous bandwidth consumptions. Consequently, conventional routing methods such as Distance Vector algorithms are not acceptable for WSNs. In addition, memory constrained sensor nodes are incapable of saving the global information of network topology as the link state algorithms do. To overcome these problems, Geographic Routing algorithms are introduced. In Geographic Routing, each node only maintains the state of its one-hop neighbors and propagates topology information of its one-hop neighbors. Therefore, in order to make forwarding decisions Geographic routing only requires the position of the packet's destination and the position of its one hop neighbors. These two properties have made Geographic Routing very assuring. All Geographic Routing protocols have two modes of operations. In first mode, they greedily forward a packet to a neighbor, which is the best choice toward the destination. "Best" is defined according to the routing algorithm; it could mean the closest node to destination or a node with highest energy level, etc. If the node fails to find the best neighbor, the algorithm switches to the second recovery mode. Different Geographic Routing protocols have different backup recovery processes.

1.2 The Problem

Commonly, large numbers of sensor nodes are required to cover an area. Therefore, nodes must be cheap to make use of the network economic. For this purpose, wireless sensors are made in small sizes, battery powered and memory constrained. They can communicate over a restricted area, as their radio range is small. Like any other broadcast oriented wireless technology, WSNs are vulnerable to numerous security threats. In addition, they are often deployed in unattended, unreliable environments where physical security is unavailable. These restrictions have made secure routing challenging in WSNs. Different attacks are applicable on different layers of WSNs. Hardware oriented attacks such as eavesdropping, interference, and jamming attack can disrupt the physical Layer operations. MAC layer attacks cause selfish misbehavior of nodes and gaining unfair share of bandwidth. Attacks such as selective forwarding, Sinkhole attack, Sybil attack, etc, target the network layer and routing protocols. Attacks such as SYN flooding, session hijacking, etc. lead to malfunctioning at the transport layer. Finally, some of the possible application layer attacks are viruses, worms, spywares, and Trojan Horses. They can attack both operating systems and user applications.

1.3 Contribution

Geographic Routing protocols have been extensively studied in ad hoc and wireless sensor networks. However, many of these protocols have not paid much attention to security at the design phase. All concerns are focused on overcoming the inherent constraints of WSNs. In this study, we attempt to analyze attacks on two unsecure Geographic Routing protocols namely GEAR and GPSR. The reason for choosing these two from a variety of unsecure Geographic Routing protocols is comparing an energy aware protocol to a distance centric protocol in an attack-wise manner. Some attacks target the routing algorithm on path selection based on the distance metric, some target the energy consumption level of the sensor nodes and some take advantage of both metrics. In this research, the impact of different types of attacks are studied, attacks such as routing misdirection, selective forwarding, Sybil attack, sinkhole attack, Byzantine attack, beaconing attack, etc. The goal is to improve the robustness and security degree of the two aforementioned protocols. To achieve this, various approaches have been tried. One approach is disjoint multipath routing in which every compromised path has a probable uncompromised alternative path. Another approach is to make the protocols trust-aware. This means forwarding decisions are made not only based on the routing algorithm but also based on the history of system behavior. Nodes that have been treated loyally in the recent past are rewarded while the suspicious malicious ones are punished. Finally, we present that if we want to secure the protocols only by relying on the capabilities of regular nodes and using the software methods without additional help of auxiliary facilities such as hardware support and cryptographic algorithms, we need to keep a partial history in every node and in every sent message.

The report is organized as follows. In section 2, we describe background information about concepts, characteristics and challenges of WSNs and Geographic Routing protocols. We also describe some of the possible attacks at the network layer. Section 3 explains about the work that has been done on the topic of this research up to this moment. In section 4, we analyze the problems in the scope of GEAR and GPSR. We try to discuss the faced challenges and how to design approaches that can overcome these challenges. Section 5, explains about the implementation of the application. In section 6, we study the simulation environment, threat models and countermeasures against described threats. Section 7, summarizes all the findings of this research and proposes additional possible work. Finally, section 8 and 9 list additional information and conclude the report with references and abbreviations.

2. Background

Concepts, Definitions, Classifications and Characteristics

2.1 Wireless Sensor Networks

A WSN [1] is a network of hundreds and thousands of small, low power, low-cost devices called sensors. A sensor is an object that performs the sensing task and converts all forms of energy into electrical energy. The core application of WSNs is to gather information about physical objects or areas and report events. Physical property that should be monitored identifies the sensor type. For example if we want to monitor temperature: Thermistors or thermocouples sensors are required. Pressure gauges sensors are for monitoring pressure, Accelerometers are used for monitoring motion vibration, etc. Some examples of WSN usage are military and civilian domain, robotic landmine detection, battlefield surveillance, environmental monitoring, wildfire detection, and traffic regulation, life-saving operations, vehicle tracking, structural health monitoring, economic forecasting etc. Since the transmission ranges of sensor nodes are not large, they cannot transmit their data directly to the base station. Therefore, they cannot form a star topology. Thus, multi-hop communication is more common for sensor networks in which, sensors form a mesh topology and every sensor node serves as a relay for other sensor nodes. This reduces the power consumption and allows for larger coverage, however, it introduces the problem of routing: the task of finding a multi-hop path from a sensor node to a base station, which is one of the most important challenges in WSNs.

2.2 Challenges in Wireless Sensor Networks

As described above, there are many challenges and constraints in WSN technology, which leads to the design of protocols and algorithms differ from other type of distributed systems. Some of the challenges are as follows:

Energy constraints

Most of the time it is not possible to replace the batteries of sensors and when the battery is depleted, the sensor is discarded. As a result, sensor nodes should be able to operate during their entire mission time with the initial installed battery [1].

Self-* properties

Since there is no possibility of maintenance and repair in remote areas and harsh environments, sensor nodes must be self-managing which means they must be able to autonomously configure themselves, cooperate with other nodes and accommodate to failures and environmental changes without human intervention. For example, sensors, which monitor the catastrophe areas or battlefields, are thrown out of the airplanes over the target areas. Not all of these sensor nodes survive intact after such a drop and may never start their sensing activities. However, those who are survived must start a consecutive setup and configuration process autonomously including determining their positions, communicating with neighboring sensor nodes and the initiation of their sensing responsibilities. A sensor node should be self-organized, which means that it should be able to adapt configuration parameters based on system and environmental situation. For example, a sensor device can change its transmission

power to communicate with more or less number of neighboring nodes. Self-optimization is the ability to monitor and optimize the use of the limited system resources such as battery by automatic sleep and wake up. Self-protection is the ability to detect and protect from attacks and intrusions. Finally, self-heal property allows sensor nodes to autonomously recover from network disruptions [1]. One form of self-healing is Self-stabilization. "A system is called self-stabilizing if and only if, regardless of the initial state and regardless of the privilege selected each time for the next move, at least one privilege will always be present and the system is guaranteed to find itself in a legitimate state after a finite number of moves [2]." A system turns out to be in an inconsistent state for many reasons. The temporal violation of algorithm assumptions, the variation of memory content due to harsh environments affects message loss at a higher accepted rate, topology changes due to node depletion, node destruction, mobility or new joining nodes, all can lead to temporal inconsistencies. The manual reconfiguration of large scale WSNs is not possible in order to recover from all these inconsistencies. Therefore, self-stabilization is a much-demanded property for algorithms in WSNs. The system assumptions could be violated not only due to the benign faults but also by a powerful adversary disrupting the network functionality. It is difficult to anticipate all possible states of a large-scale network after such an attack. Self-stabilization makes sure that the network can recover from any state as long as the assumptions hold once more.
Typically, algorithms are either secure or self-stabilized and not often secure and self-stabilized at the same time. Researches [3], [4] have been aimed at achieving high level networking protocols for WSNs that are both self-stabilizing and secure and could stand up to both faults and attacks. For example a secure and self-stabilizing algorithm can work based on this assumption that the part of the lost messages, including messages that are lost due to benign collisions or messages that are lost due to attacks do not pass a certain threshold in order to keep the adversary undetected. Therefore, the adversary does not attack all messages of a node. However, if the adversary passes this threshold, even a self-stabilized algorithm does not guarantee to provide the expected level of service. If the adversary is detected and eliminated and the message delivery assumption holds again, the self-stabilizing algorithm recovers quickly and delivers the promised level of service.

Decentralized Management

In large scale WSNs, sensor nodes cannot work in a centralized manner in which a Base station supervises the routing process and topology changes. Instead, sensor nodes must collaborate with each other locally, without a global knowledge. This brings all the challenges associated with decentralized systems such as self-learning via flooding and overhead of prompt reaction against topology changes [1].

Design Constraints

All the mentioned constraints affect the design of network protocols, operating systems and middleware in WSNs. For example, TelosB devices only have10 Kbytes of RAM and 48 Kbytes of flash memory, therefore, the installed operating systems must have small footprints and must be efficient in its resource management tasks. Alternatively, a list of neighbors is stored in a sensor node. Respectively, any algorithms that may require more computational power and storage capacities than can be provided by low-cost sensor nodes are not desirable [1].

Security

Like all other secure computer systems, WSNs need to address three well-known services in the CIA security model: Confidentiality, Integrity, and Availability. Here, we explain what does this mean in the scope of WSNs.

Confidentiality: requires the protection of the exchanged data from insider, passive adversaries. The most common methods to achieve confidentiality are cryptographic schemes such as encryption which requires efficient key management schemes [1].

Integrity: The receivers in WSN should be able to recognize whether the exchanged data between the two parties has been changed. In addition, the integrity service should also make sure that the exchanged content is not deleted, replicated, outdated or maliciously injected [1].

Availability: Due to many threats to the WSN, some portion of the network or some of the functionalities or services could be temporarily damaged or unavailable. E.g., some sensors could die prematurely. Hence, availability services make sure that the necessary functionalities and services are always up and running, even in presence of malicious nodes. Availability properties of WSNs is studied in form of Denial-of-Service type attacks [1].

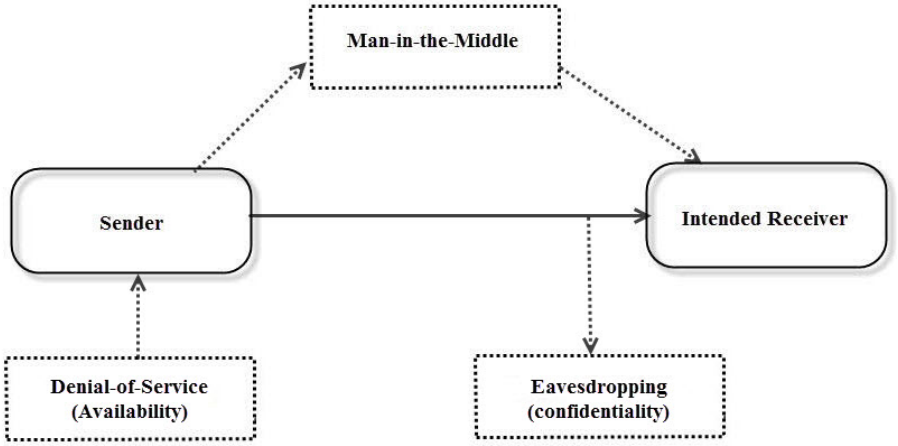

Figure.1. The CIA model and examples of attacks.

Resource constrained sensor nodes cannot support conventional security algorithms that are heavy in terms of computation or those, which require negotiation and authentication with remote devices [1]. Meanwhile, despite the fact that cryptography provides integrity, confidentiality and authentication, it is defenseless against an internal adversary. Therefore, a security mechanism inside a WSN is required to deal with internal adversaries. Wireless sensor nodes typically operate in remote and hard to reach locations, deployed in public access environments and often scale large. Thus, it is not possible to continuously monitor and protect sensor nodes from attacks. The wireless medium of communication is error prone. Errors such as channel errors, routing failures or collisions hence, packets may be lost or corrupted in the middle of the way. By the way, it is difficult to distinguish benign communication errors such as node and link failures from malicious behaviors [1]. Wireless

communications make it easy for an adversary to eavesdrop on sensor transmissions leads to one of the most challenging security threats called Denial of Service attack with aim of disrupting the availability of WSN operation. DoS attacks can be applied in different ways. As an example, powerful wireless signals jam the communication channels and prevent legitimate nodes from successful communication [1].

2.3 Geographic Routing Protocols

In wired networks, routing is performed by routers, high performance devices that specifically designed for the purpose of forwarding messages. The underlying network is stable and network topology does not change frequently. Therefore, prompt reactions and rapid propagation of update messages in short intervals are not required. In contrast, in WSNs, there are no specific devices for routing process. Every node acts as a router, cooperates on forwarding a message from source to destination. Wireless sensor networks are inherently more dynamic than the wired networks as network topology can be changeable. Thus, conventional routing protocols, which are designed for wired networks, generally fail to satisfy the requirements of wireless networks. These facts lead to invention of new routing protocols specifically for operation in ad hoc networks. Typically, these protocols are classified into three categories: proactive, reactive and Geographic Routing protocols. The first two are called topology-based protocols, while the third one is a location-based protocol.

Proactive routing protocols need to maintain the information about the entire network topology and propagate frequent updates due to topology changes. On the other hand, reactive routing protocols need to discover the routes on demand via excessive flooding. Since the topology-based protocols are beyond the scope of this research, we suffice it to say that they are not very efficient in lightweight sensor nodes for the aforementioned reasons as they exhaust the network bandwidth and need intense memory storage. However, the interested readers are referred to reference such as [5] for more information.

Alternatively, in geographic routing protocols, forwarding decisions are made using geographic position of the nodes. All nodes in the network know their own positions as well as the position of all their neighbors. In addition, every sent packet carries the location of the final destination in its header. The node uses these three inputs to choose the next hop. All routing decisions are made locally based on internal node state and therefore, very little routing information is kept in each node. Traffic overhead and computation time are considerably reduced because no energy is spent on frequent route discovery, route request and reply messages. Node memory requirements are decreased, as there is no need for keeping information about the entire network topology. These three advantages, no need for keeping routing tables, independence of remote topology changes and flooding free route discovery process, are three main reasons for the appropriateness of Geographic Routing for WSNs. This makes it so practical, as once the position of the destination is known, all operations are local [5].

2.3.1 Geographic Routing Algorithm

The formulation of above definition is as follows:
"Let G = (V, E) be a Euclidean graph. The task of a geographic ad hoc routing algorithm A is to transmit a message from a source S $\in$V to a destination D $\in$ V by sending packets over the edges of G while complying with the following conditions:

- All nodes v $\in$V know their geographic positions as well as the geographic positions of all their neighbors in G.
- The source S is informed about the position of the destination D.
- The control information which can be stored in a packet is limited by O(log n) bits, that is, only information about a constant number of nodes is allowed.
- Except for the temporary storage of packets before forwarding, a node is not allowed to maintain any information [5]."

There are various types of Geographic Routing protocols. Some examples are GPSR, GEAR, GAF, GHT, GOAFR, GFG, etc.
Before going further, we need to explain about the two main approaches of packet forwarding in Geographic routing; Greedy forwarding and Face routing.

2.3.2 Greedy Routing

The first introduced approach to Geographic Routing was greedy forwarding [5]. This approach is conceptually simple and the implementation is easy. Initially, with respect to the routing algorithm, the concept of being "best located neighbor" needs to be defined. This can be interpreted as "closest to the destination" or "having the highest energy level", or the combination of both, etc. Then every node forwards the message to its best-located neighbor for example to a node, which has the minimum distance to the destination. Greedy forwarding advances until it reaches a node without any "better" neighbor. It then stops. This dead-end which is one of the conventional problems of all Geographic Routing protocols is called "local minima", "local maxima", "hole" or "void". Consequently, a backup forwarding approach such as Face routing is required for escaping from it. We will explain about it in the following section. As a result, greedy forwarding cannot be used solely as a solution for geographic routing. However if not stuck, it reaches the destination so efficiently.

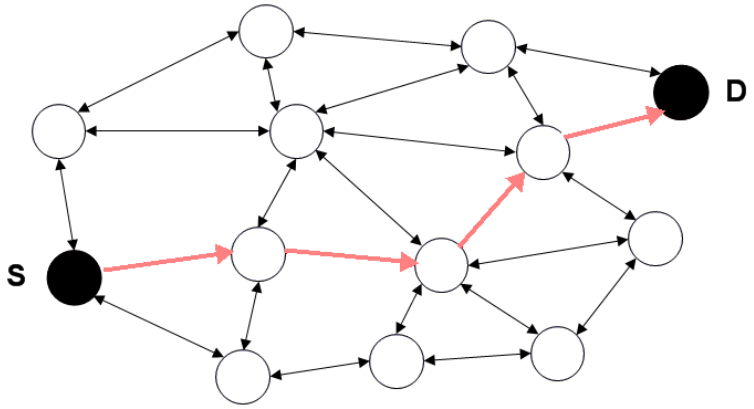

Figure.2. Greedy Routing.

If d is the distance between source and destination, it reaches the destination with cost: O (d^2).

2.3.3 Face Routing

Face routing sometimes called perimeter forwarding was the first Geographic Routing algorithm that guaranteed successful message delivery without getting stuck in the middle of the way [5]. It is defined based on the concept of "faces", contiguous polygonal regions separated by the edges of a planar graph. In a planar graph, no two edges cross each other. Face routing uses two principles of "the right hand rule" and "face change" to proceed. The right-hand rule goes around a face in a cycle on clockwise direction. There are some minor differences between protocols in how to explore the face and where to switch faces. The original algorithm keeps track of the points where it crosses the line SD (the hypothetical line that connects the source S and the destination D). After routing the face completely, the algorithm returns the intersection point that is closest to the destination. It then proceeds by routing the next face closer to D. The same steps are repeated until the message either reaches a face containing the destination or reaches a node that is no longer considered as the local minima. It then falls back to greedy mode and moves forward. As mentioned it always guarantees delivery, however if n is the total number of nodes in the network, it takes at most O (n) steps, which is similar to flooding. Thus, it cannot be used as the main routing approach and always should be used as a backup method of greedy forwarding [5] [8].

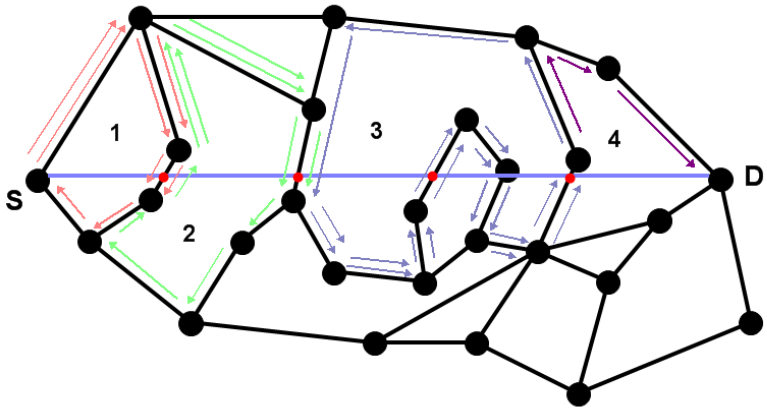

Figure.3. Face Routing.

Original face routing: if n is the total number of nodes in the network, it takes at most O (n) steps, which is similar to flooding

Current Geographic Routing protocols use one of the following rules for changing the face:

First intersection: As the algorithm is traversing the face using the right-hand rule, on encountering an edge that crosses the hypothetical line SD at a point closer to D than the point that the current face was entered, the algorithm changes the face at that edge. Thus, this approach does not traverse the entire face (GFG, GPSR) [7].
Best intersection: Works like the first intersection approach, with the difference that it traverses the entire face. (AFR, Compass) [7].
Closet-node (other face routing): This approach also traverses the entire face and changes the face at the "node" closest to D. (GOAFR+) [7].
Closest-point (other face routing): The operation is the same as closet node (other face routing), however, the face change occurs at the closest "point" to D. [7].

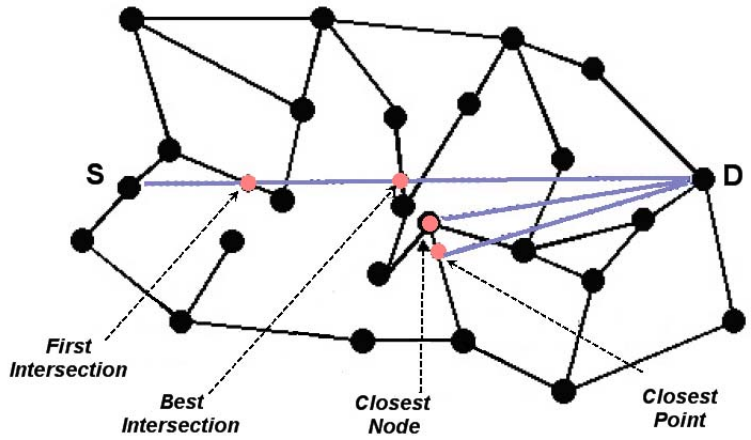

Figure.4. Points where the face change is performed.

2.4 GEAR and GPSR

In this research, we attempt to study and compare the performance and security of two Geographic routing protocols called GPSR [5] and GEAR [9]. After describing the fundamental concepts; now we are able to explain their functionality in more detail.

2.4.1 GPSR: Greedy Perimeter Stateless Routing

For advancing the packet toward its destination, GPSR uses two phases of message forwarding, greedy forwarding and face routing. As earlier described in detail, in greedy forwarding, each node looks up into its neighbor table and forwards the message to the one with closest distance to destination. When the message reaches local minima, it enters the face routing mode. For face-traversal, GPSR uses right hand rule. For face change, it uses first intersection approach. If SD is a hypothetical line, connecting source S to destination D, the message only walks faces that are cut by the line SD and only to the intersection point. The algorithm keeps the track of points where it cuts the line SD. When it reaches an edge that cuts line SD at a point closer to the destination than the point where the algorithm was switched to the current face, it performs the face change. GPSR performs well in sparse networks. It also guarantees delivery [8].

2.4.1.1 The algorithm

1. Greedy route to the target destination until a closer neighbor to the destination exists.
2. If a local minima is occurred, face route around the void, meanwhile in every taken step it is checked whether a closer neighbor is available to the destination.
3. Fall back to greedy as soon as a closer neighbor to the destination is reached.
4. Repeat step 1 to 3 until the final destination is reached.

2.4.2 GEAR: Geographic and Energy Aware Routing

In order to deliver messages to the destination, GEAR uses the energy metric for neighbor selection in addition to the distance metric. The main objectives are increasing network lifetime and balancing energy consumption among nodes. As for the forwarding approach, GEAR uses regional forwarding which limits the query flooding by sending messages only to a certain region rather than the entire network. For every region, each node maintains two costs: 1) an estimated cost, which is a combination of node's distance to the target region and node's residual energy with this assumption that there is no local minima along the path toward the destination. 2) A learned cost: This is the propagated cost after encountering the local minima and leaving it behind.
Initially, every node calculates the estimated cost: C(Ni,R), of Neighbor i: Ni, to region R according to the following formula:

$$C(Ni,R) = \alpha.d(Ni,R) + (1 - \alpha).e(Ni)$$

Equation.1. Estimated Cost Calculation

α is a tunable weight between [0,1].

After node received learned cost from neighbor i, the learned cost value of the neighbor is updated to:

$$h(Ni,R) = C(Ni,R) + h_{new}(Ni,R)$$

Equation.2. Learned Cost Calculation

As described earlier, local minima, occurs when all closer neighbors to the destination are dead. When there are no local minima throughout the path, learned cost and estimated cost are equal. However, in presence of holes, the learned cost is larger than the estimated cost because of the algorithm need to circle around the hole. In order to forward a message toward a region, nodes try to calculate the estimated cost of each neighbor. Then the packet is forwarded to a node with the minimum cost value. If routing encounters a local minima GEAR tries to circle around the depleted nodes by picking one of the neighbors with smaller learned cost value. It oscillates between alternative paths with regard to both distance and energy metrics. As a result, battery consumption is shared between several paths and nodes on a single path will not be exhausted. In addition, the calculated cost value is broadcasted to all neighbors and if there is a considerable change between the old learned cost value and the received one, the new cost value is propagated further, until all the neighbor nodes know about the new cost value. When messages reach the target region, two situations arises: number of nodes in the target region is higher than a certain threshold, then the target region is divided into n subregions and n new copies of the message is created and is sent to every subregion. This division is repeated recursively until number of the nodes in the target region is lower than a certain threshold, then recursive forwarding is stopped and GEAR applies restricted flooding by flooding the message inside the region. If the sub region is empty, the packet is dropped. The method is not efficient in sparse networks. After successfully reaching the destination, the learned cost value will be propagated upstream. It is important to know that the converge cast of the learned cost value does not affect successfully routing a packet out of the holes. It only affects the efficiency, as it gives every node the chance to know about the local minima soon enough to avoid it. Therefore, not every node needs to repeat the whole recovery process by itself. It relies on the new received learned cost value.[9]

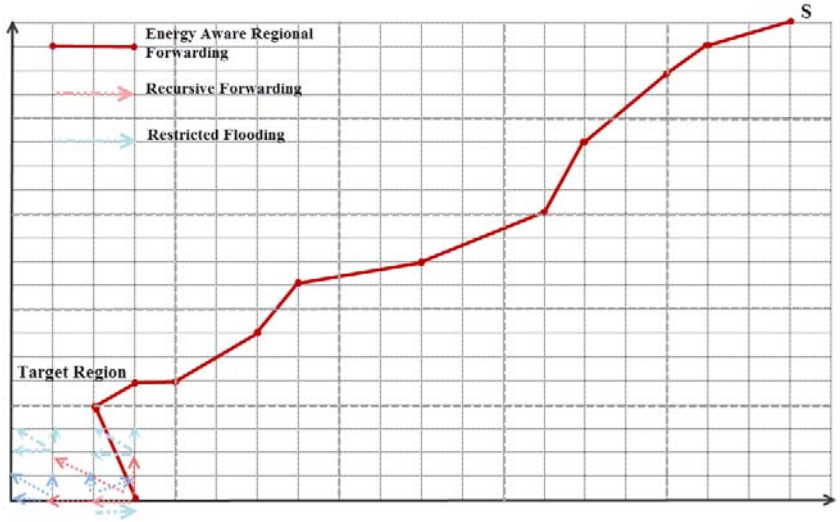

Figure.5. Geographic Energy Aware Routing.

2.4.2.1 The algorithm

GEAR algorithm has the following steps: [9]
In order to deliver a message to the target region, GEAR uses geographical and energy aware forwarding. One of the following situations may occur:

(a) When a closer neighbor to the target region is available, GEAR chooses a neighbor among all neighbors that is closest to the target region.

(b) When all neighbors are further away, GEAR has reached local minima. It then chooses a node that minimizes the tradeoff between the remaining energy level and the distance to the target region.

When the packet arrives the target region:

a) If the number of nodes inside the target region is greater than a certain threshold, GEAR uses Recursive Geographic Forwarding.
b) If the density is low inside the target region, GEAR uses restricted flooding. [9].

2.5 Attacks at the Network Layer

As previously described, the goal of a secure routing protocol is to ensure confidentiality, integrity and availability of the network traffic. Generally, attacks on different layers try to break one or all of the CIA properties and respectively a security policy for a system may demand fulfillment of one or all of these properties. Since they are nearly stateless and do not keep any history or information about the entire network topology. However, attacks on path selection such as routing misdirection or data forwarding phase attacks such as sinkhole attack or selective forwarding attack are very powerful and can cripple the functionality of Geographic routing protocols.
In this section, we describe damaging attacks on geographic routing protocols in detail. In section 4.2, we will describe how and to what extent these attacks are applicable in GEAR and GPSR. Finally, we propose a mitigation technique for each attack.

2.5.1 Terminologies

Before going further, we need to explain some terminologies:
Adversary: An adversary [16], is an entity that is trying to circumvent one of the CIA properties of the system security.
Adversaries can be classified depending on the membership of the group of nodes under study:
Internal Adversary: Is an authenticated member of the group under study. It means it has the same privileges and authenticities as the normal nodes. Attacks that are mounted by internal adversaries are more dangerous and harder to detect.
External Adversary: Is an unauthenticated node from outside the group under study.

Another way is to classify adversaries is based on their affects on the routing protocols:[16]
Active Adversary: Apply changes on the system like altering the contents of the packets, removing packets or changing the normal behavior of routing algorithm. Thus, the threat mounted by an active adversary is easier to detect and mitigate.
Passive Adversary: Does not apply any changes on the system. It just tries monitoring the data and inspecting the system's behavior by listening and eavesdropping. Since it does not leave any track, it is harder to detect and mitigate.

Another distinction can be made based on the power and resources an adversary possesses.
Mote class Adversary: The adversary has uniform capabilities as the legitimate nodes do. E.g., equal CPU, equal battery power, equal memory, etc. and therefore can affect a few sensor nodes.
Laptop class Adversary: The adversary is equipped with more powerful resources such as infinite battery power, faster CPU, longer radio transmission range. Thus, it can be more effective.

In the succeeding paragraphs, we describe attacks that we will implement in more details to provide a more clear insight.

2.5.2 Data Forwarding and Path Selection Phase attacks

The focus of this research is to generate this class of attacks. The adversary tries to place itself in the path of data flow to drop, replay or modify the data packets with the aim of compromising the integrity and availability of the network. Of this category, we implemented Selective forwarding attack, sinkhole attack, routing misdirection attack and Byzantine attack. In the following section, we describe how these attacks work in general and how to detect and mitigate them.

2.5.2.1 Selective forwarding attack

As the name implies, in selective forwarding attack, the malicious nodes refrain from forwarding certain messages and will drop them selectively. There are two types of selective forwarding attacks:
Black hole attack: Was introduced in [10]. In this attack, the adversary treats like a black hole and removes all received messages. This attack however takes the risk to be recognized easily by its neighboring nodes. As the neighbors sense there is no message exchange with a certain node they will consider it as dead or malicious and decide to seek another route. One purposed solution to avoid this attack is giving a unique sequence number to every sent packet. Therefore, the receiving side can detect the gaps and will know some nodes are dropping packets in the middle of the way[10]. Another way to detect this attack is defining a threshold; every node should have a specific rate of data flow. If the node does not meet this threshold, it is revealed that the node is misbehaving. We will use both techniques to detect the attack.

Grey hole attack: The adversary forwards or removes received packets conditionally [14]. For example, it may drop a certain percentage of received packets or it may drop certain types of packets. Since the adversary disobeys in a selective way, this attack is difficult to detect. If the adversary behaves smart enough to remain within the threshold value of trust, the detection becomes even more difficult [16].
Multipath routing [22] can be used to avoid these types of attacks. When, one path is bogus, packets route through another probable secure path. This way is also more reliable if the primary path includes disconnecting nodes. There are two mechanisms for multipath routing:
Disjoint multipath: Is composed of completely distinct and parallel paths. The routes do not cross each other and there is no intersection between the paths. As a result, failures on one path do not affect the others.
Braided multipath: The disjoint multipath approach is not always energy efficient, since the alternative path might be longer than the primary path consuming more energy. In braided multipath, which is a more complex yet, more efficient approach routing paths are not completely distinct and there are nodes, which are included in several paths.
We use disjoint multipath technique in section 2.5.2.3, to overcome the applied selective forwarding and routing misdirection attack on GEAR and GPSR.

2.5.2.2 Sinkhole attack

With the aim of compromising the availability of data in WSNs, the adversary tries to attract almost the entire traffic using the routing algorithm. Sinkhole attack is more effective in routing protocols that advertise for certain information such as the remaining energy or the location of the nearest node to the destination, in specified intervals. As an example, in GEAR nodes frequently advertise update messages about their remaining energy level. To apply the attack, the adversary advertises a false high-level value of energy. This way, other nodes are deceived and choose the adversary as their next minimum cost energy neighbor and will forward it, their data packets. On success, this attack is one the most serious attacks as it targets almost all traffic of the network. Especially when the adversary is geographically close to the final destination, the delivery ratio almost reaches zero.
After attracting the traffic, the adversary can carry out selective forwarding by dropping a part or all of the attracted packets.
In order to prevent this attack, we use the reputation and trust based systems [21]. Reputation and trust are two critical factors for decision-making in WSNs. Reputation means the opinion of one entity in the network about the degree of trustworthiness of another. Trust is the expected behavior of one entity in the network from another one.
As previously described, insider adversaries bypass the cryptographic protection mechanism and enter the system. While they enter, they have all the rights and authorities as legitimate nodes do. In such cases, conventional security and authentication mechanisms are not sufficient to protect the system. Thus, the system needs to study the past behavior of the system to analyze the reputation and forms the trust based on the collected result. It then uses trust for further routing decisions. Such a system is called reputation and trust based system.
Many reputation/trust based systems have a component called Watchdog/Path-rater [15] which monitors the neighbors by overhearing or statistical analysis of sent/received messages.
In order to share information between the nodes, the reputation/trust based systems use one of the following approaches:
Friends list: Just shares the positive information.
Blacklist: Just shares the negative information.
Reputation table: To share the positive and negative information together.
All reputation systems are prone to false positives. It is possible that nodes are identified as misbehaving nodes by mistake, either because of inaccurate observation, bogus ratings, or because of a fault in the reputation system. Thus, there should be a mechanism as compensation to frequently check whether the reason of penalizing is resolved. If so, the banned node is rejoined to the network.
Finally, all reputation-based systems have award and penalize approaches based on good and bad behavior of the entities, a misbehaving node is isolated from cooperation with other nodes.

2.5.2.3 Routing Misdirection

The adversary does not obey the routing algorithm and diverts the received messages to the wrong node instead of the optimal one. This requires the messages to go through more hops and a longer path before they reach their destination, thus consume more energy. However since this is applied on one hop only, after going through a few hops, the messages find their way back to the optimal path again. Hence, routing misdirection cannot be very effective when it is applied on one hop only. In the following section, we show another form of this attack, which is the result of collusion of multiple nodes, called byzantine attack. This way the colluding adversaries will not let the packets return to their best path.

2.5.2.4 Byzantine Attack

A byzantine attack [18] is another attack that falls under the category of data forwarding attacks and targets compromising of the availability property of WSNs. In this attack, a group of compromised nodes, located between source and destination collude with each other to disrupt the normal behavior of data forwarding. The disruption could be due to routing loop generation or deviating the packets to non-optimal paths. This causes the degradation of delivery ratio, resource exhaustion and reduces the network lifetime. If adversaries are able to cover a wide area the network throughput nearly reaches to zero. This attack is cannot merely be detected using the watchdog and path-rater. However in [20] an efficient protection method is purposed. In this approach if a node's percentage of data flow does not meet a specified trusted threshold, Forerunner packets are sent along the path to notify other nodes about the existence of malicious nodes. This way other nodes are notified about the bogus path and gradually isolate it. In section 6.4, we use this method for breaking routing loops.

2.5.3 Impersonation attacks

Impersonation attacks try to violate the Confidentiality property of CIA properties. An adversary can masquerade as other nodes and initiate many attacks in the network. For example, by acquiring the MAC address or the IP address of a legitimate node the adversary can misrepresent its identity. A Sybil attack is a perfect example of such type of attacks [11][12], [13]. Bellow we describe the Sybil attack and how this can be applied and to what extent it can cause damage. In section 6.2, we apply Sybil attack on GPSR and GEAR and we will try to provide a solution to minimize the damage.

2.5.3.1 Sybil attack

In this attack, one node pseudonymously generates a large number of identities to mislead the legitimate nodes into believing that they have many neighbors. Compared with other forms of attacks, Sybil attack requires very little in the way of specialized hardware or cooperation with other nodes to cause its intended damage. This attack is especially more effective on systems which need to reach consensus and agree on a decision value or on any group of nodes which need to collaborate with each other to achieve a common goal. Some examples of such systems are multipath routing, where all disjoint paths are under the control of the adversary and its Sybil identities. Geographic routing, where the adversary target the underlying

localization scheme and presents multiple coordinates and claims to be in more than one place at once[17]. In section 4.3.1, we describe how the mentioned attack affects GEAR and GPSR. "Some other vulnerable systems are data aggregation, voting algorithms, fair resource allocation and misbehavior detection [17]."

There are several approaches to detect and mitigate Sybil attack. Hardware oriented approaches, cryptographic approaches and neighbor analysis. The first two methods are briefly explained in section 3; in the following section, we describe the third method, which is our applied mitigation approach against Sybil attack.

Neighbor Analysis:[18]this approach works based on analysis of the neighboring information of each node. Some of methods we describe above need special hardware support, and some of them are centralized approaches. We are seeking a decentralized hardware independent method that works in uniform environment when all nodes are equal and have the same capabilities. The cryptographic approach cannot do anything about situations when the adversary manages to break the cryptographic guard or simply being an internal adversary. Therefore, neighbor analysis seems to be the most possible approach.

This method take advantages of this fact that all of the Sybil nodes have a similar set of neighbors because they are all associated with the same physical node. Therefore, there should be a set of nodes, which appears more often in the neighbor set of other nodes. The objective of the detection scheme is to find such a set, which is called critical set, CNB. The Sybil nodes are verified through this set. Before going further, we need to describe the following notations[18]:

"SN: The set of all nodes, including the Sybil nodes that have been forged by the adversaries.

M: The adversary: the node that applies the Sybil attack.

NB_i: the set of i's neighbors, $i \in SN$.

$CNB_{i,j}$: the set of common neighbors for both i and j; $i, j \in SN$; $i \neq j$. Therefore,

$CNB_{i,j} = NBi \cap NBj$.

NB^n_i: the set of i's legitimate neighbors, $i \in SN$. Since NB_i contains all of i's neighbors, including Sybil nodes and normal nodes, $NB^n_i \subset NB_i$.

$CNB^n_{i,j}$: the set of common normal neighbors of both i and j. Also, $CNB^n_{i,j} \subset CNB_{i,j}$.

Assume that legitimate node V gets suspicious notices that it has not received sufficient number of messages from its neighbor in the recent rounds and initiates the detection process. In order to find C, V first establishes $CNB_{i,v}$, $\forall\ i \in NB_V$. The procedure employed to determine each $CNB_{i,v}$ is as follows:

1- Node Node V broadcasts a request message to one of its neighbors, e.g. node i.
2- When i receives this message, it broadcasts a message over its maximum transmission range.
3- Any node hearing this message (e.g. node j) replies using one-hop broadcast directly to V.
4- Node V records the IDs of the nodes which send a reply and combines these IDs to form the $CNB_{i,v}$.
5- The above steps are repeated until all of the $CNB_{i,v}$ have been collected.

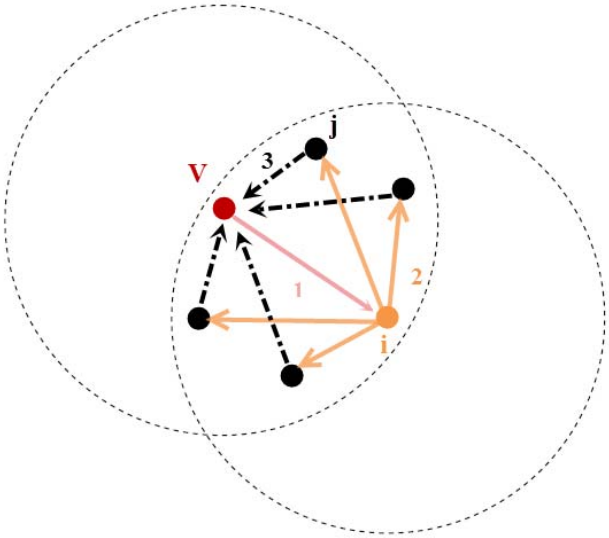

Figure.6. Node V attempts to collect intersection of all of its neighbors.

In this approach, the detection method works based on a simple information collection and analysis approach rather than direct examination of every node. This strategy prevents the malicious node intentionally responds with false information in order to deceive V. In the second step, when i broadcasts, not only it is heard by all its neighbors, but also by node V itself. If V does not hear the broadcast from i, it is clear that i is an adversary/Sybil node. This prevents the adversary from hiding its legitimate neighbors. After collecting all the information from its neighbors, node V will count the number of times that each node appears in the $CNB_{i,V}$ $\forall$ i $\in$ NB_V compiled by node V. The number is defined as AP(i) which means the total number of times that the node i appears in $CNB_{j,V}$ $\forall$ i $\in$ NB_V. Since the Sybil nodes have the same set of neighbors and the appearance of Sybil nodes is much more than the legitimate nodes, there will be some nodes whose AP(i) is much higher. Therefore, it turns out that these nodes are the neighbors of Sybil nodes. If the number of appearances of a node i exceeds a certain threshold value θ, the node is designated a critical member and assigned to C. Hence,

C $\{i|AP(i) > \theta$ i $\in NB_V\}$. Node i is considered as a Sybil node if C $\subset CNB_{i,v.}$

θ determines the size of the set C that has a direct impact on the efficiency of the detection method. Thus, if node V which starts the detection process has a total number of |N| neighbors (including legitimate and Sybil nodes), the value of θ is around 0.79|N| [18]."

2.5.3.2 Hybrid attacks

In all previously described cases, attacks can be combined in many different ways. There are two main reasons for this. First, the attack is intensified and can harm more effectively. For example, a selective forwarding attack can combine with sinkhole attack. The adversary first attracts the traffic then it removes the packet.

Second, it can remain more obscure and the detection will be more difficult.

3. Preliminaries and Related Work

In recent years, there have been many papers describing security challenges of WSNs. (C. Karlof and D. Wagner.2003) [16] present an extensive survey of attacks against different routing protocols of WSNs from different perspectives. For Geographic routing protocols, they introduce attacks on the underlying localization scheme and attacks against data forwarding phase. In the final section of the paper, they propose some general countermeasures against the examined attacks. However, the described attacks and the proposed countermeasures are very general and not in detail, i.e. they are not related to specific Geographic routing protocols. A thorough and comprehensive study is done by (T. Roosta et al.2006) The paper elaborates attacks on all layers of WSNs specially attacks on the application layer and underlying operating system (TinyOS). However, in the same way as [16] the study is very general and not detailed.

(J. Dong et al. 2010) [28] have done an innovative research about the security of the common localization system of all Geographic routing protocols and how this is affected by a various types of attacks. Some of the examined attacks are coordinate deflation, inflation and oscillation, in which the adversary attempts to violate the location precision and constancy properties by causing legitimate nodes to obtain incorrectly small, large or fluctuating coordinates. The paper also evaluates the effect of two other attacks on coordinate request and coordinate reply messages by violating the availability and accuracy of the gained destination coordinates. The authors call the attack coordinate pollution that drops or modify the correct replies. A similar research by (N. Abu Ghazaleh et al. 2005) [6] investigates the misbehavior of nodes, which report false location in beaconing phase. Additionally, the research considers the mobility attack when the adversary changes its location immediately after reporting it. Although both papers are very comprehensive and complete, research has tended to focus on security of the localization technique rather than the vulnerabilities of the routing algorithms themselves.

Many works on the security of sensor networks have focused on attacks on secrecy and authentication via proposing key management schemes. The approach relies on some kind of cryptographic public key that is shared between the receiver and the transmitter. The purpose of key management is to generate, install, update, distribute, revoke and store the associated keys through a trusted third party providing the key management service. The Certificate authority is responsible for the public key distribution as well as revocation of the key in case it is compromised. However, the lack of centralized infrastructure makes public key authentication systems less effective. In order to solve this problem, especially in GPSR (M. Erritali et al. 2011) use symmetric cryptography by adding a digital signature to sent packets through the AES algorithm and the MD5 hash function [29]. However, although the measurements are decentralized, they are restricted to impersonation attacks, replay attacks or any modification and replication of routing information. They provide no solution against path selection or data forwarding attacks. Meanwhile, digital signatures are too heavy in terms of computational power of a CPU limited sensor node. In addition, the code itself is too much due to memory limitations.

(D. Liu et al. 2004) [30], attempt to avoid the problem of heavy digital signatures by introducing μTESLA (the “micro” version of the Timed, Efficient, Streaming, Loss-tolerant Authentication Protocol) which solves the impractical inadequacies of other authenticated broadcast protocols by using only symmetric mechanisms for authentication instead of signing the initial packet with a too expensive digital signature. In addition, instead of the energy consuming approach that includes a key in each packet, μTESLA includes the key once per period. Moreover, since it is costly to store a chain of one-way keys in a sensor node, the numbers of authenticated senders are restricted by μTESLA. In a complementary paper

(Perrig et al., 2002) [31] introduce SPINS (Security Protocols for Sensor Networks). SPINS includes two protocols: SNEP (Secure Network Encryption Protocol) and μTESLA. SNEP provides confidentiality and mutual data authentication. SPINS assumes that each node shares a master key K with the base station. All the other keys, including encryption key, MAC generation key, and random number generation keys are derived from the master key using a string one-way has function. SPINS uses RC5 protocol for confidentiality. SNEP exchanges the counter, to keep the counter value at both sides synchronized. Since SNEP increments the counter value after sending each message, the message is encrypted in a different way each time. It also provides data authentication in which a receiver can verify that the message originated from the associated sender if the MAC verification produces acceptable results. SNEP also avoid replay attacks using the counter value in the MAC. The counter acts as a nonce and prevents the adversary from replaying old messages. However, keeping a counter in every endpoint, SPINS decreases the communication overhead.
Various frameworks have been proposed for link-layer security in WSNs. TinySec[32] developed by (C.Karlof et al 2004) is a security architecture as a part of the official TinyOS release. For authentication, TinySec has two operation modes: The authentication only mode, which is the default mode and every packet, is adjusted with a MAC using Cipher Block Chaining Message Authentication Code (CBC–MAC). However, the payload is not encrypted. In the authenticated encryption mode, first, the payload is encrypted and then the MAC is computed. TinySec provides all the CIA properties. Its default block cipher is Skipjack and it operates based on the CBC–CS mode. The short key length of Skipjack is likely make the cipher unsecure in the near future and CBC–MAC has security deficiencies. The semantic security is obtained through an 8-byte initialization vector and the 2 extra bytes of the counter per packet is small. The total overhead of energy, latency and bandwidth is less than ten percent.
Contiki [33] developed by (Dunkels A. et al 2004) is another operating system for WSNs. Although it is highly memory efficient, portable and multitasking it does not support data confidentiality and authenticity. It only offers integrity through Cyclic Redundancy Checks. ContikiSec [34] designed and implemented by (L. Casado and Ph.Tsigas 2009) provides all the CIA security properties for Contiki at network layer. Via performance evaluation in the MSB-430 platform, the design balances the energy consumption against the security of the Contiki OS. The performance is achieved through using of six different block ciphers: AES, Skipjack, RC5, Twofish, Triple-DES and XTEA. The focus is to speed up the encryption process and to reduce memory and energy consumption. The authors study the performance of the CMAC, OCB and CBC–CS operation modes and conclude that among all studied block ciphers, AES is the most suitable one for WSNs.
(Tanachawiwat et al., 2003) [37] have brought out TRANS: a Trust Routing for Location Aware Sensor Networks. The protocol is intended for data centric networks. TRANS uses μTESLA to provide message authentication and confidentiality. Using μTESLA, TRANS makes sure that the packet is passing through a set of trustworthy nodes and utilizes Geographic Routing. The base station broadcasts an encrypted message and only trusted neighbors are able to decrypt the message using the shared key. In the return direction, the trusted neighbors append their locations to the message and encrypt it with the shared key. The message is then forwarded to the closest neighbors toward the destination. On receiving the message, the source node (the base station with the associated MAC) is authenticated. In order to send an acknowledgement or a reply message, nodes use the same trusted path that they received the message it. The solution however relies on the base station, which makes it a centralized solution and brings out all the problems of centralized solutions.

Various security controls use intrusion detection systems for ad hoc network security. Among all the intrusion detection techniques, anomaly detection is economic in terms of energy consumption; this makes it more suitable for the resource constrained ad hoc networks.
Typically, algorithms for anomaly detection are based on iterative and block based methods. An anomaly occurs when the value of some traffic metric suddenly exceeds from its normal value. "(a. Patcha et al, 2007) propose three methods for anomaly detection: the classifier based anomaly detection, the finite state machine anomaly detection and the game approach anomaly detection. The paper categorizes these three approaches into misuse detection and anomaly detection. [35]." Misuse detection examines the system against known attacks and known weaknesses to match and identify the intrusions. Obviously, this method is limited to predefined and well-known patterns and is helpless with new attacks and unpredicted system faults. However, it is very efficient and accurate for mitigating of known attacks. Anomaly detection systems create a standard measure of normal system and any system activity that deviates from the standard is treated as an anomaly. This method does not require and is not limited to preliminary data of the well knowing intrusions. However its main drawback is that it has a high false positive rate, however since it does not store a database for attack profiles it is very suitable for WSNs. Additionally, the authors suggest two anomaly detection methods which work based on the constructing an attack graph for intrusion detection. The first method is designed based on the adjacency matrix attack graph and predicts single steps together with multiple steps attacks. The second method uses the distances attack graph in order to correlate intrusion events and to build attack scenarios and therefore is more appropriate for collaborative and dynamic WSNS, especially at the application level.
(D, Tang et al. 2010) [36], Propose a secure and energy aware (SEAR) routing protocol to optimize lifetime and security of the network. The algorithm attempts to balance energy consumption and make routing decision based on probabilistic random walking.
The random forwarding is used to create routing unpredictability for source privacy and jamming prevention. The algorithm is very robust, since security is considered in the design of the algorithm. However, with the constraint that it has assumed some random nodes are equipped with specialized hardware, which mean they would have some technical advantages over other sensor nodes. These nodes can analyze signal strength and signal direction and are capable of overhearing and monitoring the traffic of their neighbors. Moreover, although the proposed random walking makes the routing decision very unpredictable under some special circumstances it can be very inefficient for energy aware routing protocols as it make routing decisions randomly, sometimes, it may never reach destination.
(Sen et al., 2010) [38] Proposes a precautionary measure against selective forwarding attacks based on a robust neighborhood monitoring system (NMS) which is an energy aware routing protocol. The protocol identifies the malicious and faulty nodes soon enough and select routing paths which do not go through these nodes. NMS monitors the neighborhood nodes using neighbor list checking approach for detecting packet-dropping attacks. Although most of the existing algorithms use multipath routing approach to mitigate selective forwarding attacks, NMS uses a single path routing mechanism. If the packet encounters a malicious node on its way, it attempts to circle around in an efficient manner and return to the single shortest path. The new route is selected using sending broadcasts in the neighborhood of the malicious node. At neighbor discovery phase, in addition to its one hop neighbors the protocol keeps the information about its two hop neighbors i.e. neighbors of neighbors. In addition, a key is shared between a node and its one-hop neighbors. When a node sends a packet to its neighbor, it keeps a copy of the packet, encrypts it with the shared key and forwards it to the next-hop. Since the key is shared between the node and all its neighbors, all one-hop neighbors can overhear and monitor whether the encrypted packet is forwarded any further or whether it is forwarded intact. This method also relies on node additional overhearing capabilities.

Furthermore, it has the overhead of keeping of one packet copy for every sent packet. Finally, the encryption of every sent packet and its decryption by all one-hop neighbors in order to check whether the packet is remained intact is another overhead.

(J. Newsome et al. 2004) [17] have done an Innovative research on different methods of applying and detection of Sybil attack. For mitigation, the research proposes several hardware oriented approaches. One way is examining the radio strength of nodes based on the fact that all Sybil nodes are one physical node and all of them have the same radio strength. Therefore, this check reveals the true identity of the Sybil nodes. Another suggested centralized verification method works based on this fact that there should be a trusted central authority such as a base station, which knows about the identity of all the deployed nodes. To detect Sybil attack, the central authority could ask all of the nodes in the network about their identities and compare the results to the specified deployment. Although this is an effective solution to detect the attack, it has one main drawback. The list of specified identities must be protected as well from not being modified by malicious nodes. If the adversary manages to modify the list with adding its fake identities, the Sybil nodes will be added to the network. Another proposed solution in the paper is to verify the physical position of every node. As all the Sybil nodes have the same location, this approach reveals the identity of malicious nodes. The paper also suggests an approach using symmetric cryptography, in which a unique key is shared between nodes and a trusted base station. By negotiating the shared key, two nodes can verify each other's identity and authenticated and encrypted links are established between every pair of neighboring nodes. In order to defend against an insider adversary, the base station can restrict the number of allowed neighbors of a node to a certain threshold. On violation, error messages are sent. However, this method is no good for avoiding outsider adversaries.

As we have discussed, plenty of papers have done extensive researches in the subject of secure routing protocols in WSN. Some introduced new strength points however ended up with some other weaknesses themselves. Consequently, new ones came up to cover and supplement the previous ones. This is because the security of WSNs if an emerging and ongoing technology and it is very difficult to find a comprehensive solution, which covers all the problems that need to be dealt with. In our research, we attempt to find a solution for secure routing of GEAR and GPSR, which only relies on the inherent capabilities of the regular sensor nodes without equipping them with any specialized hardware. We aim for a pure software solution, independent of a separate and extra supervisor component. We try to find a fully distributed solution without reliance on a central base station. After all the explanation we have made so far, we know in a large scale sensor network, nodes can be far away from the base station and this brings the concept of multi-hop routing and as we explained in detail sensor nodes should have all the self-* properties. Therefore, we are seeking a solution based on the fact that every node is completely able to manage its security requirements without any special and additional help. To the best of our knowledge, up to this moment, no research has studied GEAR and GPSR security from this perspective. We try to focus not only on the secrecy and authentication attacks but also on path selection and data forwarding attacks, which target the availability and integrity of the system. To achieve this we focus on finding self-reliant, decentralized and software-centric solutions.

4. Analysis and Design

In previous sections, we broadly described WSN constraints, security challenges and weaknesses of available geographic routing protocols. In this section, we narrow down our study to GEAR and GPSR. We attempt to analyze the vulnerable points of these protocols and to study how the adversaries can take advantage of these weaknesses to mount the previously described attacks. Nevertheless, before we go further we need to describe our system model and our testbed assumptions.

4.1 Design Considerations

4.1.1 System Assumptions and Constraints

Our model is composed of a large number of sensor nodes forming a rectangular grid. All nodes have equal and bidirectional radio transmission range. In addition, they frequently broadcast beacon messages. This way every node knows about its neighbors and their locations. We also assume there are no obstacles and nodes that are located within the radio range of each other can always communicate. We assume that there is not any node mobility or node failures, thus, changes in the topology happen only due to node depletion. As a result, topology changes are slow comparing to the rate of advertisements and all nodes have a consistent view of the network. For not making the testbed too idealized, we use a noise simulation model, which is discussed in section 5.1.2. Each sensor node has a limited and irreplaceable energy resource and all nodes start with equal energy level. Idealized unicasts from random senders are sent to random receivers through a multi-hop routing strategy. This means the network is fully distributed and there is no single base station. Each sensor node is assigned a node ID corresponding to its location. We have a very simple localization technique based on the ID of the nodes. If X is the number of columns in the grid, the x coordinate is (ID mod X) and the y coordinate is (ID / X). The IDs are static and are not changed during the routing process. There is no special hardware control or monitoring on the system. We do not rely on node's overhearing capabilities or adjusting the transmission range. Our nodes are not armed with any special cryptographic scheme. In some simulation runs, the adversaries are randomly located. However, in some others we found some certain areas more attractive to be targeted.
Since we are not equipped with any specialized hardware or any cryptographic scheme, our main approach to detect and mitigate attacks is analyzing the traffic and the recent history of the nodes. However to achieve this goal, there is no separate supervising component such as a watchdog or a path-rater. Every node is self-reliant and operates independently through keeping a partial history about the past messages. In addition, messages carry with them a partial track of nodes, which they have passed through.

4.1.2 The Adversarial Model

We assume the adversaries have the following properties:
The adversaries are mote class adversaries. It means they are not equipped with stronger radios, faster CPUs or larger energy resources and memory. They have no technical advantages and are uniform to the legitimate sensor nodes.
All the adversaries are insiders, that means we assume they have already passed the authentication and authorization phase and are members of the network under study.
All the adversaries are active. They try to influence the routing algorithm by altering the routing path, dropping packets and so on. It is important to note, fields of the packet header

are initialized only at the origin and remain intact throughout the path. This means that, if the adversary is the origin of the sent packet, it can set the packet header with wrong values. However, it cannot not modify the header or the content of received packet. In our simulation, the only attack relies on packet modification is reporting false position attack which is applied to Sybil and routing misdetection attacks and happen at the origin of the packet not in the middle of the way.

4.1.3 Goals

In our design, we are seeking to achieve two main goals. Our first goal is to maximize the network lifetime. GEAR provides this purpose to some extent. Additionally we need to make sure that the routing misdirection and byzantine attacks are detected and avoided. As previously mentioned these two attacks are aiming for prolonging the routing process and decreasing the network lifetime.
Our second goal is to increase the delivery ratio through detecting and avoiding packet-dropping attacks. Our criterion for measuring the performance is delivery ratio, which is the number of packets that are successfully delivered to their intended destination. Moreover, we are interested in how this criterion scales with network size. As previously described in detail, sensor networks may involve thousands of nodes, thus scaling to large size network is an essential aim for routing protocols to be applicable to sensor networks.

4.2 GEAR and GPSR Vulnerabilities

If we desire a real secure routing protocol, security measures should be braided with every choice in the design phase. It would be far more effective to try to predict and keep a bad thing from happening than to fix it once it has happened. In routing protocols such as GPSR and GEAR, the protocols are not designed with security in mind. Therefore, despite all the efforts it takes to patch the protocol, the adversary can always find a new security hole to attack.
As described earlier, both GEAR and GPSR have a path selection phase in which a forwarding node independently choose the next hop according to the routing algorithm. There is, however, no special centralized policing mechanism in this phase due to the distributed nature of the algorithms. As assumed GEAR and GPSR do not rely on nodes overhearing capabilities, nodes are not aware of neighboring nodes operation. Every node should be able to stand on its own and decide for its own. Once a message leaves the node, no history is kept about it and no acknowledgement is sent back to the forwarding node about its whereabouts. We will use this rule to mount path selection attacks such as routing misdirection and byzantine attacks. Some random nodes are chosen to disobey the algorithm by acting in a self-ruling way. When they receive the packet, they forward it to any neighbor other than the “best selected” one, e.g. a randomly selected neighbor. This way the routing path is prolonged and the network lifetime is shortened. We also mount the Byzantine attack by choosing some certain nodes as adversaries to forward the packet directly to their conspirators rather than their best next neighbors. Another possible way to take advantage of this situation is applying data forwarding attack such as selective forwarding. In this case, we put some nodes to drop the received packets randomly.

As explained, update messages are broadcasted in GEAR each time a message successfully leaves a void behind. These update messages are broadcasted to let the other nodes about two things, first, a local minima is coming up, avoid it. Second, update your new learned cost

value with the cost of the local minima free path. We take advantage of these frequent updates in two ways. First, we choose some random nodes to deceive other nodes by broadcasting update messages with an ideally small-learned cost value of a local minima free path. All the receivers of this fallacious update messages choose the route that passes through the malicious node and the system end ups with a sinkhole attack.
In the second approach, the malicious node broadcasts update messages with a learned cost value, which is significantly different from the real one. All the receivers will also broadcast update messages about this new value, which causes even more update broadcasts. This leads to the overflow of node's message queues. As a result, nodes drop every incoming message and practically stop working.
As previously mentioned, in both protocols, nodes use a simple beaconing protocol to exchange location and battery information with their neighbors. Every node uses a simple localization scheme to calculate its location. It is not possible to equip all sensor nodes in the network with GPS as it is expensive, not efficient in terms of energy consumption and small sensor nodes cannot handle its size. Therefore, only a few number of sensor nodes, which are called anchors, or beacons know their locations through GPS. Regular nodes then calculate distances or angles between neighbors locally and exchange the results to calculate their location. Since none of the two protocols has a secure localization scheme, we choose some random nodes to report false location information to other nodes in their beaconing phase. First, the adversary sends beacon messages multiple times and each time with a new false location. The receiver of this beacon messages mistakenly thinks it has several neighbors. While there is only one neighbor with different IDs, this causes the Sybil attack, which we already described. The second time, the adversary reports an incorrect small distance with a powerful signal, the receiver finds it closer than real and it would affect its forwarding decision which we describe in more detail in section 4.3.2.

4.3 Applied attacks

Now that we have described the various vulnerabilities of the protocols and the capabilities of the adversaries, we can think of applying attacks, which we have introduced in section 2.5.

4.3.1 Sybil attack in Combination with Selective Forwarding

4.3.1.1 Threat model

In section 2.5.3.1, we explained about the impacts of Sybil attack. Now we are going to see how this attack will affect our system operation. As our experimental tests have shown, when random senders send messages to random receivers, it is more likely that nodes located in the center of the network send and receive more traffic. Therefore, we assume that the adversary is more motivated to target the central nodes in order to be more effective. With this aim, we try to choose the location of impersonated nodes in a way that they cover a larger number of nodes. One way is to choose a certain number of consecutive horizontal or vertical nodes. This way, the adversary makes sure that sufficient routing paths are passing through it and its Sybil nodes.

Impersonation happens at Neighbor detection phase when the every node broadcasts HELLO messages to its maximum radio range. The adversary sends a message on behalf of all of its impersonated nodes stamped with their ID in addition to its own one beacon message. Every node that receives these impersonated messages put the sender in its neighbor list and considers them in their further routing decisions. Now when nodes send messages to any of the Sybil nodes, the adversary receives the message. Up to now, the attack has minor effect; the adversary just takes the control of normal nodes and steals a small percentage of the messages. For applying a more effective attack, we combined it with selective forwarding attack in which, the adversary drops a certain number of received messages.

4.3.1.2 Detection

As we assumed we are not equipped with any specialized hardware for monitoring the channels or measuring the signal strength, also we are not able to verify the location of the nodes. Therefore, as long as the Sybil attack just steals the packets of its victims the effect remains negligible and we cannot detect it. It is when the adversary starts to drop packets selectively that we can detect the attack and react accordingly. As our statistical and empirical evidences have shown when the application is run under normal circumstances and in the absence of malicious nodes, every node meets a certain threshold of traffic exchange with its neighbors. One way to detect the attack is verifying the threshold value in every certain number of rounds. If the node finds it has not met the threshold in the specified interval, it can initiate a Sybil detection process. The detection process [18] that we described in detail in section 2.5.3.1, works based on this fact that all Sybil nodes are all associated with the same physical node and therefore share the same set of neighbors. Therefore, by collecting neighboring information and analyzing the results the initiator node detects the identity of the impersonated nodes.

In addition to message types of the original routing protocols, three message types are added in order to collect neighboring information. We discuss about these message types in detail in section 5.2.3.

Assume node-A detects the attack. It sends a message to all of its neighbors requesting their critical set. All neighbors of node-A broadcast a message and ask the receivers to send their ID to node-A. Node-A receives messages from the nodes, which are in its radio range (its

neighbors). Node-A then collects and count received reply messages. Nodes, which have sent more number of replies, are impersonated.

4.3.1.3 Mitigation

When Sybil nodes are detected, the initiator node broadcasts a message with the ids of the Sybil nodes as the content. For GEAR, receivers of the broadcast notification, set the learned cost value of adversary and its Sybil nodes to ∞ in their neighbor tables. For GPSR, a flag is set for corresponding neighbors to show that they are adversaries. This way the node is isolated from later routing decisions.

4.3.1.4 Tradeoff

As the studied topology is grid with high density the detection performs without false alarms. However, we need to keep an array in every node to collect and count the reply messages. Based on the maximum transmission range, which is 25, each node has at most eighty neighbors in our system; therefore, the array length is 80. The elements of arrays are C structs each composed of three bytes, 2 bytes for ID of the reply sender nodes, and one byte as a counter, which counts number of received replies. Overall, each node needs to keep 240 extra bytes for detection of the attack. However, storage overheard is negligible comparing to the message overhead. In order to detect and mitigate the Sybil attack through this approach we need to send many messages. First, a unicast is sent to every neighbor, second, all the receivers of the unicast message send a broadcast to their maximum radio range. Third, the receivers of the broadcast try to send a reply to the initiator node whether their replies reach the initiator or not it has the overhead of energy and bandwidth consumption. Finally, a broadcast message sent by the initiator to notify every node in the range about the identity of the adversary and its Sybil nodes. Since GEAR broadcasts frequent update messages, this extra message overhead decrease efficiency of the protocol significantly by decreasing the network lifetime, specially the node that initiates the detection process sends and receives more messages, so it causes the reveal of Sybil nodes and survive the other nodes but it dies itself quickly. Overall, the solution does not work very well for GEAR. In section 6.2, we show this inefficiency through the achieved result. Comparing to GEAR the solution works good for GPSR as the message queue is not overflowed with both update and broadcast messages and improves the performance to some extent.

4.3.2 Routing misdirection in Combination with Sinkhole attack

4.3.2.1 Threat model

In order to apply routing misdirection attack, the adversary forwards the packets to a random neighbor instead of the closet neighbor in GPSR and the lowest cost neighbor in GEAR. As we mentioned in 2.5.2.3, the one-hop routing misdirection alone is not very effective, since after disorienting through a few hops, packets return to the shortest path. Thus, we combine this attack with sinkhole attack.
The mechanism of applying Sinkhole attack is entirely different for two protocols.
For GEAR, in order to attract the traffic, the adversary uses the ID of a victim node to broadcasts update messages with a false and very small-learned cost value. This way, other nodes consider the path to the victim node as the lowest cost path and forward their traffic toward it. It is important to know that the adversary is not willing to receive the traffic itself as this causes its battery depletion very quickly. Thus, it advertises on behalf of another node. As

a result, all the nearby nodes send an avalanche of packets to the victim, the node will die and all the packets are vanished into oblivion.
For GPSR the adversary takes advantage of broadcasting HELLO messages in neighbor detection phase. In section 4.2, we described how the unsecure localization mechanism could open the way for reporting false position attacks. In our adversarial model, the adversary can report false position in two ways: In first, it uses the id of a dead node as the sender. In the second, it advertises with the ID of a node, which is not in the radio range of the receivers and very close to the destination of the packets. In both cases, the receivers of the HELLO messages consider the sender as their neighbor and involve them in their later routing decisions. When the deceived node attempt to forward a packet to this fake neighbor, either the message is forwarded to a dead node or an out of radio range node, in both cases, packets are get lost and are never delivered.

4.3.2.2 Detection and Mitigation

As previously, described using multipath routing [22] mechanism provides more security and reliability. Since routes are dynamically formed in GEAR and GPSR, there is no primary shortest path in the network to form an alternative path with respect to it. Thus, primary and alternate paths are formed in parallel based on the fact that each node can differentiate between the best and the second best neighbors. In this technique, for every original sent (content) message on the primary path, a copy is sent to the second best neighbor on the parallel path. Every node that receives the alternative message forwards it to the next best neighbor. This way the probability that both paths consist the adversary is likely low. This method can be effective to mitigate the attack.
To prevent the described Sinkhole attack in GEAR, we study the history of the system and its behavior in recent past. Since the adversary takes advantage of low learned cost update broadcasts, studying the advertised learned cost value detects and reveals the adversary. When a node advertises a new leaned cost value, this value should be reasonable in comparison with the previous one. Therefore, when the difference between the previously reported learned cost value and the newly received learned cost value through the update message is greater than a specific threshold, the new value is too good to be true and the advertiser node is declared suspicious and is isolated. For this purpose, the learned cost of the false advertiser is set to infinity. This way the node is isolated and is banned in further routing decisions.
To protect against bogus localization information in GPSR, HELLO messages received by every node are checked. As a solution proposed in [24], nodes cannot receive HELLO messages from two different neighbors if the distance between them is greater than a certain threshold. If that would be the case, one of these two neighbors is malicious. In our simulation model, the maximum transmission range is 25, if node A with coordination: (X_i, Yi) receives a HELLO message from node B with coordination (X_j, Yj) then the equation:
$(X_j-X_i)^2+(Y_j-Y_i)^2<=25$ must hold. [24] Maximum transmission range is read from the input topology file. If the equation is not hold for a sender of the HELLO message the node is considered as an adversary and is isolated.

4.3.2.3 Tradeoff

In multipath routing, the redundancy brings reliability however; we pay the price with more traffic, more energy and bandwidth consumption. As an additional packet is sent in parallel with every original packet, fewer numbers of messages are delivered. In section 2.5.2.3, we show how the performance of the system is affected by applying multipath routing on GEAR

and GPSR under normal circumstances and when there is no adversary in the network. However, the reputation-based system is a very lightweight and efficient approach. All we need to keep is a trust value in the neighbor table and a comparison statement. Therefore, in terms of computations it imposes no burden at all. This is why the delivery ratio dramatically increases, as we will show in section 2.5.2.3.

4.3.3 Byzantine attack

4.3.3.1 Threat model

With the aim of reducing network lifetime and delivery ratio, we apply Byzantine attack via generating routing loops. For this purpose, we select a certain number of nodes as adversaries. The adversaries should be in the transmission range of each other, so they can collude to prevail the attack. In section 6.4, we will show how the simulation results have proved the magnitude of the area under attack is more important than the number of adversaries. This means the delivery ratio will not change much if we increase number of adversaries, but when we are widening the area under attack with the same number of adversaries. However, the loop diameter should not be too large; otherwise, the adversary may lose the communication with its conspirators due to the unstable wireless communication. If we want to simulate a wide loop, we need laptop class adversaries with unlimited resources. In that case, the attack will turn into another form of attack called "wormhole attack" when two powerful adversaries far from each other collude together to disrupt the network topology and take the data forwarding process under control.
In order to apply the attack, the adversaries forward a certain percentage of received data packets in a legitimate way. For the rest, they keep on forwarding the packets to one another and forming a loop. We need to remember that the adversaries need to forward a portion of received data messages in a legitimate way in order to remain within the threshold of trust and not to be revealed. Circling messages around the loop causes the wastage of energy and packet loss.

4.3.3.2 Detection

To detect routing loops a partial history of the previous received messages is recorded in every node. The database should be small due to memory constraints of sensor nodes. For this reason, we implement it as a circular queue in which, old messages at the head of the queue are replaced by new messages. In addition, in order to make unique identifiers, the source node adds its id to every sent message therefore, the tuple: <source_id, sequence> is the unique id of the message. In order to discover a loop this unique id is looked up in the partial database. If the id is present in the database, it means this message has already been received and there is a loop along the path.

4.3.3.3 Mitigation

In order to break routing loops and isolate the adversaries, each node maintains a trust value TN_i per each neighbor node. TN_i is a decimal neutral value in [0, 1] initially. The trust level of every neighbor is an estimation of the probability that this neighbor correctly forwards the packet toward the destination. With detection of every misbehavior, the trust value is decreased. Here, the most important and critical issue is how to adaptively adjust the trust value parameter based on detection of misbehavior.

In order to make GEAR trust aware, every node chooses its next-hop node according to the trust value and the remaining energy level of the neighbors. For GPSR the next hop is selected based on its distance to destination and its trustworthiness:

$$h(Ni,R) = (\alpha.d(Ni,R)+(1-\alpha)e(Ni) + h_{new}(Ni,r)) / TN_i$$

Equation.3. GEAR is made Trust-Aware

$$C(Ni,R)= d(Ni,R) / TNi$$

Equation.4. GPSR is made Trust-Aware

By comparing the above cost values between all neighbors, the GEAR and GPSR identify next hop towards the destination. If a node discovers a loop, it adjusts the trust level of its neighbors.

When misbehavior is detected, not only the packet is dropped, but also trust level of the sender and the next hop is degraded. However, this degradation is unidirectional as node-B degraded trust level of node-A, but node-A also must be notified to not send a packet to node-B anymore. For this purpose, a forerunner message is sent to the sender of the duplicate packet. Since the trust level is degraded, next time on selection of the next hop the Equation.3. and Equation.4. gives a higher value and node will select another node as next hop. Here is the idea: the sender is accusable because it has forwarded a repetitive packet, the next hop is accusable because the last time we have forwarded the message to it, it has forwarded the message to a wrong node and has caused the return of the message to us again. We do not exactly know which one is responsible, the next hop, or the hops after the next hop. Reducing the degree of trust level, the loop will be broken eventually. In addition, forerunner messages with IDs of accusable nodes are broadcasted and all receivers of the broadcast adjust their trust value of the corresponding nodes.

4.3.3.4 Tradeoff

As mentioned before, one way of discovering routing loops is nodes overhearing of each other. However since we have assumed we do not rely on this capability, we need to keep a partial database in every node, a partial trace in every sent message and sending forerunner messages to other nodes in order to let them know about the misbehaving nodes. Additionally, a counter for number of received messages from every neighbor is maintained and is incremented each time a message is received from the corresponding neighbor.

The length of the partial database is selected equal to ten, since the maximum radio range is 25, the radius of the circle is equal to 5 and its diameter is 10. Hence the maximum distance between two colluding adversaries is equal to 10.Tests have shown this estimation is large enough for keeping track of every node forming a loop. The elements of the array database are C structs composed two items: of one byte for source_id, the id of the originator of the message and two bytes for the sequence number of the message.

In addition to keeping these partial history forerunner messages needed to be sent to notify the other nodes about the existence of the misbehaving nodes, However just putting them at the tail of the queue is not fast enough. Therefore, each time a forerunner message in enqueued the queue is sorted to push them to the head of the queue and give them priority over the other type of messages. This way other nodes are notified about the misbehaving nodes earlier and try to avoid them faster.

The partial trace, which is kept in every message, is to discover GEAR routing loops. As GEAR is not a loop free protocol by nature, in some situations it wastes a number of messages by generating routing loops.

In order to pick the next hop GEAR looks up the minimum cost neighbor in the partial trace in the message header. If found it means that the message has already visited the next hop and try to find the next minimum cost neighbor for avoiding the loop. As we will see in 6.4, we tried 2, 3, 4 as the length of the partial trace, however the loop just got bigger. Finally, by choosing number 5 and with the help of forerunner messages, the loops were broken.

4.4 Application Design

We are now able to design an application that meets the described requirements and objectives. The application, which is referred to as GeoRout henceforth, is composed of ten modules, each with a specific task. There are many similarities between GEAR and GPSR such as beaconing phase, CRUD neighboring related operations, etc. In the following section, we will describe in more detail how to benefit from these similarities to create a parameterized application, which reduces the code redundancy.

4.4.1 GPSR and GEAR Generalisation

In Equation.1, we showed how GEAR makes its routing decisions. If we use the following settings, the formula can be used for both GEAR and GPSR.

GEAR	GPSR
$\alpha = 0.5$	$\alpha = 1$
Region length = 4	Region length = 1
Region width = 4	Region width = 1
Density threshold = 4	Density threshold = 1

Table.1. Parameter setting for GEAR and GPSR.

By setting $\alpha = 1$, in $C(Ni,R) = \alpha.d(Ni,R) + (1 - \alpha).e(Ni)$ formula, the energy awareness is turned off and the formula is turned into a pure distance centric algorithm usable for GPSR: $C(Ni,R) = d(Ni,R)$. As we know GEAR forwards regionally. Density threshold represents that the target region can be recursively subdivided to 4 subregions. However, in GPSR there is no regional packet forwarding. Messages are sent from a single sender to a single receiver and we have greedy forwarding instead of recursive forwarding. Therefore, the density threshold is set to 1 means that, there will be no regional subdivision.
In addition, by setting all region length and region width values to 1, we convert regional data forwarding of GEAR to normal data forwarding in GPSR.
After a common module is designed for both protocols, it is possible to test the protocols under exactly the same circumstances and thence we will be able to compare the performance of two protocols. In section 6.1, we will see practically how the energy aware decision-making differs from a pure geographic distance centric routing scheme, both under normal situation and under attack.

4.4.2 Modules Description

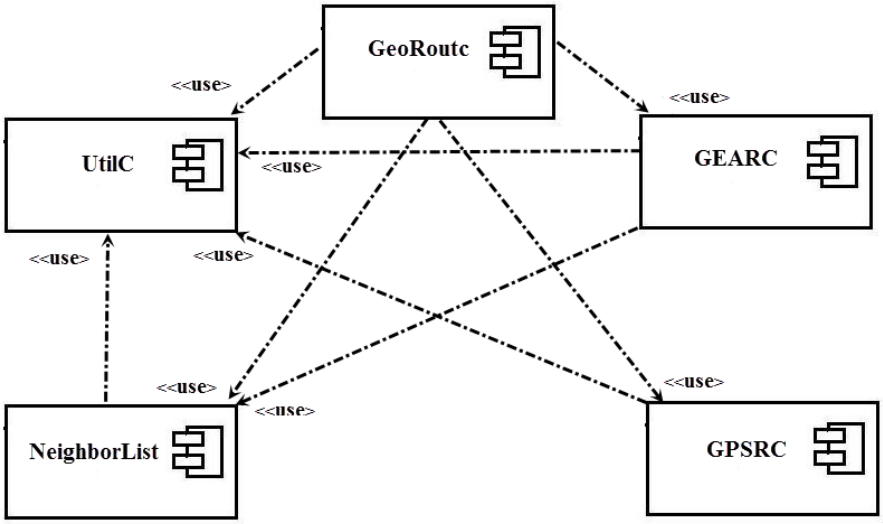

Figure.7. GeoRout component diagram.

Figure.7. depicts the generalization model of GEAR and GPSR. As shown, GeoRout is the assemblies of five user-defined modules. In addition, the application uses TinyOS built-in modules, which we will describe in more detail in section 5.2.1. We now describe the operation of each module in detail in the succeeding paragraphs.
GeoRout, which is one of the shared modules between the two protocols, is the main module and the heart of the application. The module triggers the run of the application by starting the timers and booting and initialization of the system based on the static compile time switches and the associated topology file. Sending and receiving messages over the radio and message queuing are also handled in GeoRout module. In the beginning, every node broadcasts HELLO messages on its maximum radio range. Every node that receives the HELLO message creates its initial neighbor state about the sender as its one-hop neighbor. Another responsibility of this module is to check the energy level of the nodes. If the battery level reaches to zero, GeoRout stops the timer, which means the node will stop functioning. GeoRout is also responsible for clock synchronization and concurrency control between the nodes.

NeighborList is another shared module between the two protocols and is responsible for the tasks related to inserting, updating, searching and deleting of state of the neighboring nodes. For GPSR, NeighborList finds next greedy hop or next counterclockwise perimeter hop. For GEAR, the module calculates the estimated cost, learned cost and finds the minimum cost neighbor for regional forwarding.

GPSR performs the GPSR specific operations. In original GPSR, the module forwards the message to the next hop in the shortest path. In case of multipath routing, it calculates both: next closest hop and the second closet hop for the alternative path. When routing reaches local minima this module handles switching from greedy mode to perimeter mode and performs the

face change. Respectively when the local minima is recovered the module returns the running mode from perimeter to greedy.

GEAR module, performs operations that are specific to the GEAR protocol. The module forwards the packet to the minimum cost neighbor node for regional data forwarding. If the application is run under multipath routing, the second minimum cost neighbor node is calculated for the alternative path. If the packet reaches the target region GEAR module performs either the recursive forwarding or restricted flooding. Based on the calculated learned cost value via NeighborList module, update messages are prepared in this module and are broadcasted via GeoRout module.

Util module provides all the mathematical functions, which are not built-in in nesC, such as computing the cosine, absolute, slope; determinants, etc. It also handles some utility functions such as distance calculation and range checking.

5. Implementation

In section 4, we discussed the problem, the requirements and the solution from a high-level standpoint. In this section, we describe the discussed material in more detail. Before we begin, we need to define a series of hardware, software and technology terms.

5.1 The Environment

We use tinyos-2.1.1 as the underlying platform, which is installed on an Ubuntu 12.04.1 LTS machine. Since TOSSIM works for TinyOS-based WSNs, we need to choose TOSSIM as the simulator. Respectively, we choose MICAz platform as currently this is the only platform, which is supported by TOSSIM. More precisely, the choice of TinyOS brings the other choices with it. In the succeeding paragraphs, we will elaborate more about the reasoning behind these choices and we will explain how these choices affect our application implementation.

5.1.1 The Hardware

Our simulated network is composed of MICAz sensor nodes. Since we intended to test the application for large-scale networks (e.g. composed of 2800 nodes), it was impractical to use real MICAz sensor nodes and therefore we choose TOSSIM to simulate the real environment. However as we will repeatedly refer to MICAz capabilities and features in the following sections, it is good to give a brief description about it. MICAz's RF transceiver complies with IEEE 802.15.4 standard. The CPU power is 7 MHz and the data rate could be up to 250 kbps. MICAz supports Direct sequence spread spectrum radio which makes it resistance to jamming causes inherent data security. It also supports hardware security (AES-128). DSSS shares the channel between multiple users and reduces noise level including signal and background noise. MICAz supports the TinyOS operating system. The development of custom sensor applications is specialized for battery-powered networks.

5.1.2 The Simulator

As mentioned, it is often impractical to test protocols for real networks consisting of thousands of nodes. Therefore, simulation tools play a significant role in research and development on sensor network applications. Since different types of sensor networks are different from each other in many different ways, it is critical to choose an appropriate simulator. As we chose TinyOS as the underlying platform, we needed to choose TOSSIM [26] for a simulator. TOSSIM scales to thousands of sensor nodes and its advantages include its scalability and extensibility. It simulates events directly from TinyOS components and runs the same code on all sensor nodes. Low-level components such that hardware interrupts are replaced with events and the interrupts are delivered, which compel the execution of a TinyOS application. Furthermore, TinyOS code runs in the simulator unmodified. TOSSIM works at the bit level instead of packet level, which means, an event is generated for each transmitted or received bit. Therefore, this allows simulation of applications for both low-level protocols and high-level protocols.

5.1.3 Network topology

To run the application, it is necessary to specify the network topology as an input parameter for TOSSIM. As mentioned earlier, the nodes form a rectangular grid. If X is the number of columns in the grid, nodes are numbered from 0 to (X-1) on the first row, X to (2*X -1) on the second row and so on. For example for a 20*20-grid, the ids of the first row are from 0 to 19, second row from 20 to 39 and 380 to 399 for the last row. To build topology files we use Python langue. "The *add (source, destination, gain)* method, is used to add a link from source to destination with a certain gain. When *source* transmits, *destination* will receive the packet that made weak by the *gain* value. The default value for radio model of TOSSIM is based upon the CC2420 radio, which is used by the MICAz family. For wireless noise simulation model, we use CPM (closest-pattern matching)[39]." This model, [40] provides a more accurate software simulation environment by taking advantage of time correlated noise characteristics.

5.2 GeoRout Software Requirements and Architecture

5.2.1 The Operating System

We use tinyos-2.1.1 as the underlying platform. TinyOS [26] is an operating system and a run time environments for WSNs. Comparing to its counterparts TinyOS is used more widely. The architecture consists of a scheduler and a set of components, which can be connected with each other through well-defined interfaces. Components are classified into configuration components and modules. Modules are the basic building blocks of a TinyOS program and configurations specify how two or more modules are wired up together. A TinyOS application is produced from a combination of multiple configurations into a single executable code. Like Objects in Object Oriented Programming, a component encapsulates a state and interacts through well-defined interfaces. Components communicate with each other through events and commands. Higher-level components call commands of the lower-level components and implement the event handlers. An interface can define commands and event handlers. Resource allocation in TinyOS is static. This means the required memory of the application is known at compile time and dynamic memory allocation is not possible. In addition, the scheduling policy is non-preemptive scheduling, which causes minimal memory requirement [1].

5.2.2 Programming Language

The application is programmed in nesC [26] language, in which, the development of sensor software focuses on a per-node level. The sensing application is described as a collection of pair wise interactions of individual sensor nodes. NesC is an extension to the C programming language and provides language constructs to implement code for motes and must address the unique challenges of WSNs. For example, sending and receiving a message may occur simultaneously with processing of the data and sensor nodes must be able to concurrently perform their processing tasks and responding to events. NesC implements common and utility functions as built in libraries, therefore it reduces programmers work and saves their time. As another advantage, the nesC compiler prevents concurrency bugs through a built in race condition detector [1].

5.2.3 Software Architecture

GeoRout is implemented in nesC for TinyOS and evaluated on MICAz sensor motes. Figure.7. shows an architectural overview of the application. In nesC, call paths are defined statically. Every binding between callers and callees take place at compile time. Every module can have a corresponding interface including its relevant commands. Commands, which are actually a set of named functions, provide the interactions between a provider component and a user component. The interaction is a function call and the "user component", calls the commands of the "provider component". As described in 5.2.1, modules are implemented in nesC and configurations wire them up together. Two components can communicate with each other only if they are wired up together by a configuration file. Figure.8 shows the interaction between GeoRout module and TinyOS built in modules. As shown, module GeoRout uses Main module through Boot interface, module ActiveMessage through Receive interface and so on.

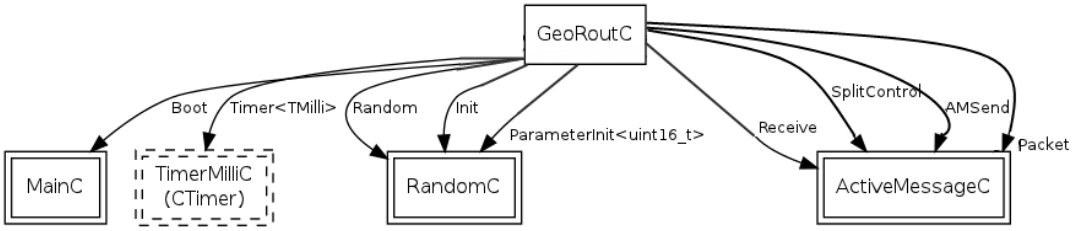

Figure.8. The intercommunication between module GeoRout and TinyOS built-in modules.

Figure.9. shows the interaction between GeoRout module and the user-defined modules which are described in section 4.4.2. The dashed rectangles represent the interfaces along with their associated modules. As can be seen, some components such as NeighborList and Util are multiply wired and are used by more than one component.

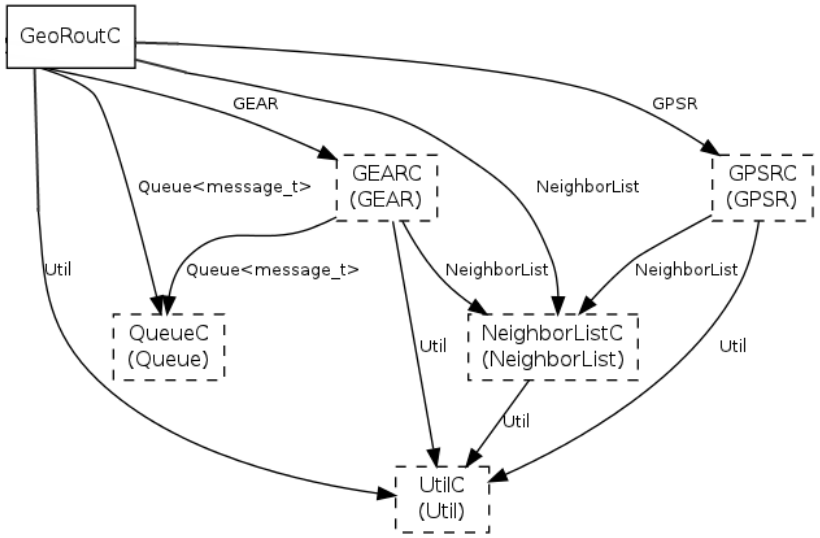

Figure.9. The intercommunication between module GeoRout and user-defined modules.

Table.2. lists the files, which are the implementations of the modules. Constants and used C structs are defined in header files. The run mode of the application is chosen via configuration parameters in Makefile at compile time. There are two possible switches _GEAR and _GPSR. Our complete implementation of GeoRout application composed of almost 4000 lines of nesC code.

Configuration	Interfaces	Modules	Headers
GeoRoutAppC.nc	GEAR.nc	GEARC.nc	GeoRout.h
	GeoRout.nc	GeoRoutC.nc	GPSR.h
	GPSR.nc	GPSRC.nc	NeighborList.h
	NeighborList.nc	NeighborListC.nc	NetworkConfiguration.h
	Util.nc	UtilC.nc	Util.h

Table.2. Implemented files for GeoRout application.

Different message types are referenced around the report, here in table 3 we describe about their concept and operations.

Message Type	Module	Description
CONTENT	Common	Data message
HELLO	Common	Beacon message
UPDATE	GEAR	Update battery & learned cost
DISSEMINATION	GEAR	Data message
QUERY	GEAR	Data message
SUB_TYPE_GREEDY	GPSR	Data message
SUB_TYPE_PERIMETER	GPSR	Data message
FORERUNNER	Common	Mitigate routing loops
REQUEST_CRITICAL_SET	Common	Sybil Detection
REQUEST_NEIGHBOR_SET	Common	Sybil Detection
REPLY_NEIGHBOR_SET	Common	Sybil Detection
ATTACKER_IDS	Common	Sybil Mitigation

Table.3. Message types.

CONTENT messages are data messages. They contain one byte of payload data.
HELLO messages are beacon messages, which are broadcasted in neighbor detection phase.
UPDATE messages, which are specific to GEAR protocol, are broadcasted after a significant difference is detected between the old and new learned, cost value. Furthermore, each time GEAR successfully recovers the local minima the UPDATE messages are broadcasted to let the other nodes know about the path which contains the hole, this way they will rely in this updated information and will not get stuck in the local minima themselves. DISSEMINATION messages are created when the CONTENT message reaches target region and tries to do the Recursive forwarding operation. QUERY messages are created and broadcasted when the DISSEMINATION messages find that there are fewer nodes in the current subregion than the threshold and try to do the restricted flooding. GREEY and PERIMETER messages are specific to GPSR and are used for greedy and face routing.

FORERUNNER messages are created after making protocols trust aware, they are sent to make the knowledge of the nodes of each other bidirectional and let the other nodes know about the existence of the loop and who is responsible for it. REQUEST_CRITICAL_SET messages are sent to neighbor nodes of a node who detects the Sybil attack. The neighbors of the Sybil attack detector broadcast REQUEST_NEIGHBOR_SET messages. These messages are broadcasted in request of listing the neighbor sets. REPLY_NEIGHBOR_SET are messages, which are replied to the detector of the Sybil attack. This way the detector can collect and count number of replies and decide accordingly. ATTACKERS_ID messages are broadcast messages, which are sent to every node in the transmission range, in order to notify them about the existence and the identity of Sybil nodes.

Message field	size	Protocol
Type	1	Common
Source_id	1	Common
Subtype	1	GPSR
From	2	Common
final_dst	2	GPSR
next_hop	2	Common
Seq	4	Common
Content	1	Common
battery_level	2	GEAR
target_region	2	GEAR
learned_cost	4	GEAR
Lp	2	GPSR
Lf	2	GPSR
edge0 edge1	2 2	GPSR
counter	2	Common
passed_nodes[5]	5×2	Common
source_id	2	Common

Table.4. Format of GeoRout messages.

Neighbor state	Size	protocol
neighbor_id	2	Common
battery_level	2	GEAR
region_postalCode	2	GEAR
Region	2	GEAR
number_of_received_messages	2	Common
trust_level	2	Common

message_database {
source
seq
}
	10*6	Common

Table.5. Maintained state in each node.

The format of messages in GeoRout application is shown in Table.4. “type” presents one of the listed type in table 9. “subtype” is specific to GPSR protocol and shows whether the routing algorithm forwards the content messages in greedy mode or perimeter mode. ”source_id” is the id of the originator of the message and is immutable, once it is set in the origin; it will not change along the path. “from” is marked with the id of the sender and is updated while getting through each hop. “final_dst” is specific to GPSR and is set at the source node with the final destination of the message and remains constant through forwarding over the network. “next_hop” is the calculated best next hop in both GEAR and GPSR and the message is forwarded to it. “content” is a 1 byte payload which is every message carries. “seq”, is the sequence number of the message increased by one in every round. “battery_level”, shows the energy level of the node, decreased by 1 in each send and receive in GEAR and is based on distance in GPSR. “target_region”, is set with the target region of the message in GEAR, “learned_cost”, the calculated learned cost in GEAR. Lp: Location where packet entered perimeter mode face routing. Lf : Point on the hypothetical

line that connects source and destination, which packet enters the current face, e0 one vertices of the first edge traversed on current face, e1, the other side vertex of the first edge traversed on current face. passed_nodes[5], every message carries the history of its last passed for the purpose of loop discovery. The id of the source of the message, is set at source node and is never are updated. “hop_count”, number of nodes the message has passed through, it is increased by one with every hop is visited on the path. Table.5. shows the state that each node maintains about its neighbors: “neighbor_id”, the id of the neighbor. “battery_level”, the energy level of the neighbor. “region”, the region of the neighbor. “number_of_received_messages”, number of received message from the neighbor. “trust_level” is the trust value of neighbor, which is updated by each loop discovery. “message_database”: keeps a partial history of the last 10 received messages in order to detect network loops.

6. Simulation Results

The application is simulated and tested based on the specifications listed in Table.6. Some of these values are obtained after several experiments and are chosen because they have shown results that are more logical. All of the nodes are considered stationary. To ensure the validity of the results, each simulation was repeated 10 times and the results were then averaged to obtain a final value.

Parameter	Value
Radio range	25 units
Data payload	1 byte
Packet size	Non trust-aware: 31; Trust-aware: 47
Packet Queue length	100 packets
HELLO period	1 millisecond
Traffic duration	280 seconds
Initial trust value	1
Initial battery powers	1000 unit
Battery consumption	1 unit per Send/Receive
Number of nodes	400, 600, 800,1200, 2800, 4800
Number of Senders/Receivers	10 random Senders/Receivers per run
Number of simulation round	10 sequential run of 78 rounds
Region	Grid of {20×20, 30×20, 40×20, 40×30, 80×60} Units
Density	High at the beginning, getting sparse gradually

Table.6. Simulation parameters.

6.1 GPSR and GEAR before Attack

As we described in section 4.4.1, we choose the same application with same input parameters for both protocols. For performance metric, we choose the number of data messages that are successfully delivered before the battery of all the nodes run out. This metric indicates the network lifetime.

As can be seen, GEAR delivers almost 30% more packets than GPSR. The reason is, GPSR considers only the distance metric and always uses same nodes for routing around the holes in perimeter forward mode, and therefore, those nodes consisting the face around the local minima are depleted sooner. This way, nodes that form the direct paths between source and destination die sooner. However, GEAR considers both distance and remaining energy metric and oscillates between different paths for leaving the holes behind. As a result, in GPSR, nodes comprising the perimeter faces die quickly and divide network into partitions. Therefore, GPSR delivers less number of messages.

Figure.10. shows number of delivered messages under normal circumstances, when there are no malicious nodes in the network.

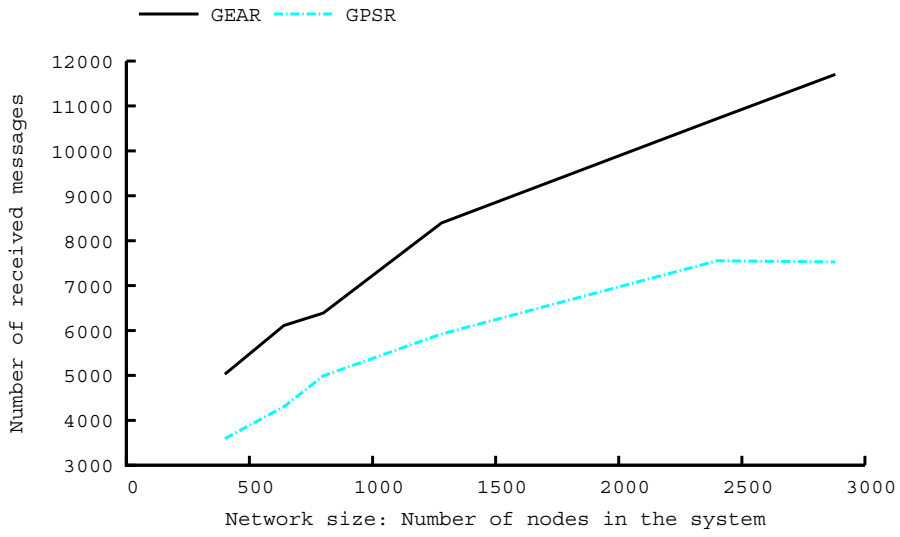

Figure.10. # of received message in GEAR and GPSR before attack.

6.2 Sybil Attack in Combination with Selective Forwarding Attack

In order to model Sybil attack with described scenario in section 4.3.1 profile, we choose a central node as the adversary to impersonate five horizontal consecutive neighbors to the right and five horizontal consecutive neighbors to the left, five vertical neighbors atop and five vertical neighbors below. This way, the adversary makes sure that sufficient routing paths are passing through it. After combining the attack with selective forwarding attack to the Sybil nodes drop more than 50% of the received packets in a random way. The results are shown in Figure.11.

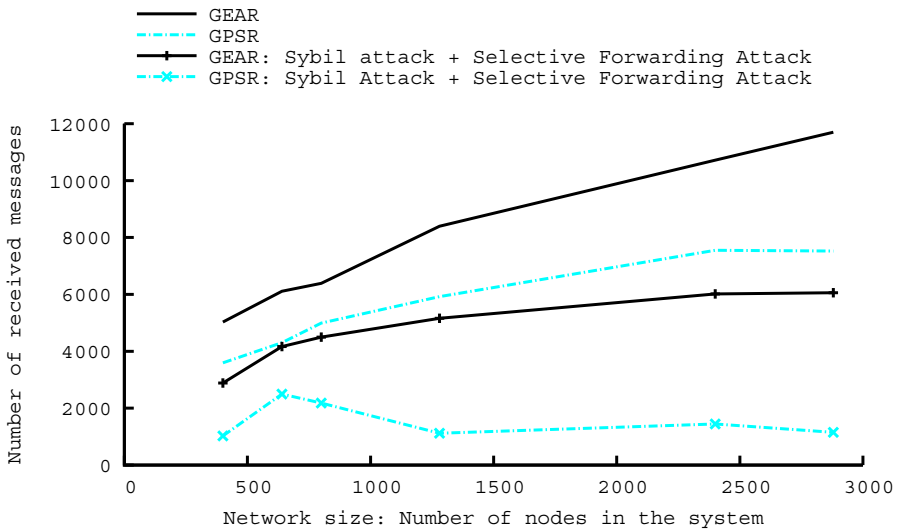

Figure.11. # of received messages under Sybil attack and Selective forwarding attack.

We previously described that our main approach to detect attacks is statistical and empirical evidences. Our tests have shown when the application is run under normal condition and absence of malicious nodes, every node receives at least one message from one of its neighbors in every five rounds therefore, each node keeps a counter per neighbor and increments it every time it receives a message. Each node verifies the counter in every five rounds and when it finds that it has received no messages from its neighbors in this interval, it initiates a Sybil detection process, otherwise reset the counter.

We previously described about the overhead of Sybil detection process. As can be seen through the results this method is not very efficient in GEAR. It decreases network lifetime and specifically the lifetime of the initiator node. Different types of messages are exchanged between nodes and all are enqueued in the same message queue. Situation arises when the dequeuing message rate does not match with enqueuing rate, the queue overflows and drops arrived messages. Therefore, we needed to exceed the message queue size to 150 in order to achieve results for GEAR as shown in Figure.12. With this change, the performance is improved to some extent. However, this does not overcome the energy consumption problem.

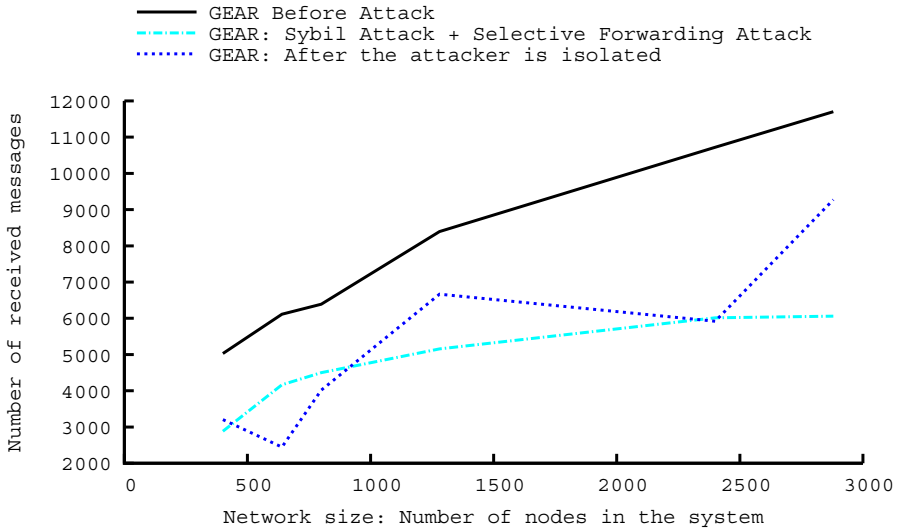

Figure.12. GEAR: # of delivered messages after applying the protection method.

As shown in Figure.13. GPSR shows a better performance due to not overflowing message queue and sending less advertising messages.

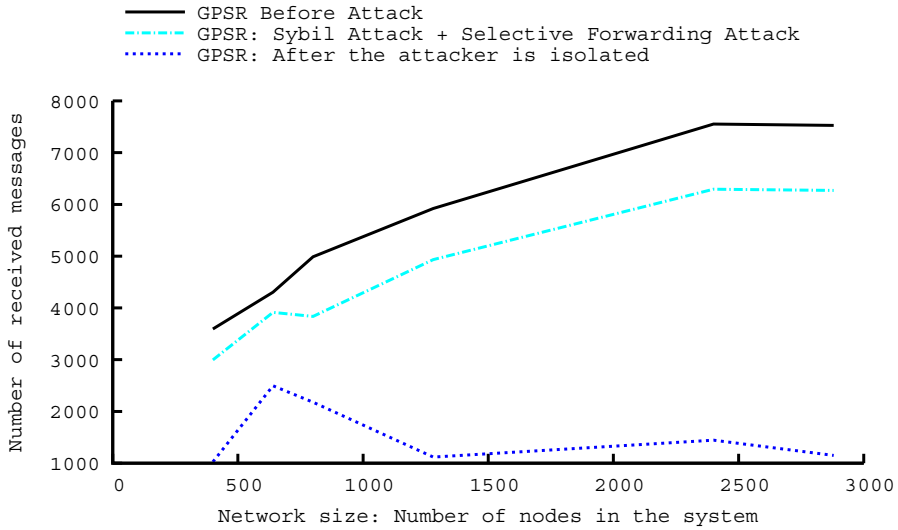

Figure.13. GPSR: # of delivered messages after applying the protection method.

6.3 Routing Misdirection in Combination with Sinkhole Attack

In order to apply the described scenario in section 4.3.2, a different percentage of nodes are chosen as adversaries. Figure.14. depicts the result of one-hop routing misdirection. As can be seen the deviation of packets from their shortest path just for one hop is not effective. If 10 percent of the nodes are adversaries, the system performance is reduced only by 20 percent. In an unrealistic scenario, when the entire nodes are adversaries, still, the delivery ratio does not reach zero. The reason is, although the adversary forwards the packet to a random neighbor instead of the best one, there are some nodes among the neighbors, which are equally the best choice and have the same cost toward the destination and the random choice by the adversary is equally good with the algorithm best choice.
As an objective example, in a network comprised of 400 nodes we need to put 40 nodes to play the role of adversaries, if we desire 20 percent delivery ratio decrease, which is not very pragmatic. Typically, an attack should have the ability to apply its intended harm with a much smaller number of adversaries. In section 6.4, we will turn routing misdirection attack into byzantine attack in which a packet will be deviated from the shortest path for more than one hop and will be far more effective.

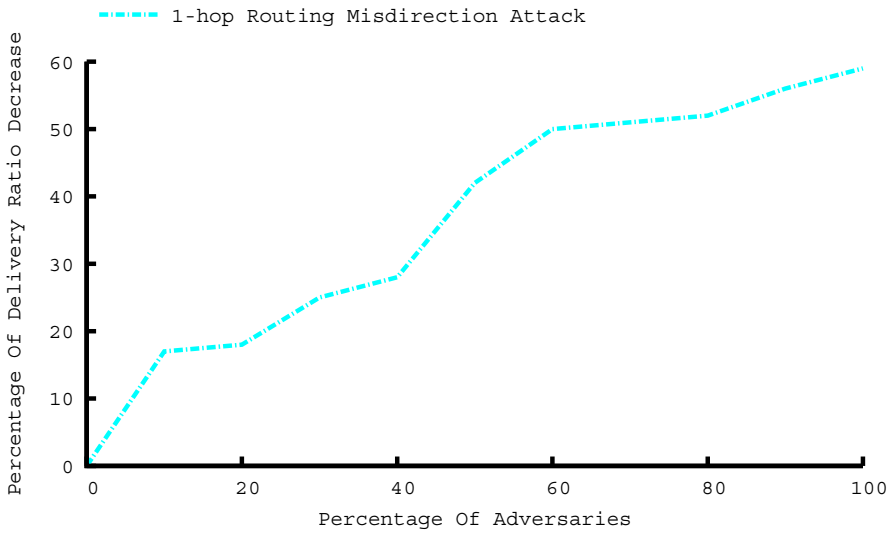

Figure.14. Delivery ratio reduction under 1-hop Routing Misdirection attack.

In order to make Sinkhole attack more effective we chose the adversaries near to the target region. Our results have shown that, the network performance is decreased significantly even with very small numbers of adversaries. As Figure.15. shows, if 1 percent of nodes are adversaries the delivery ratio decreases up to 40% and with 15% number of adversaries, the delivery ratio reaches to zero.

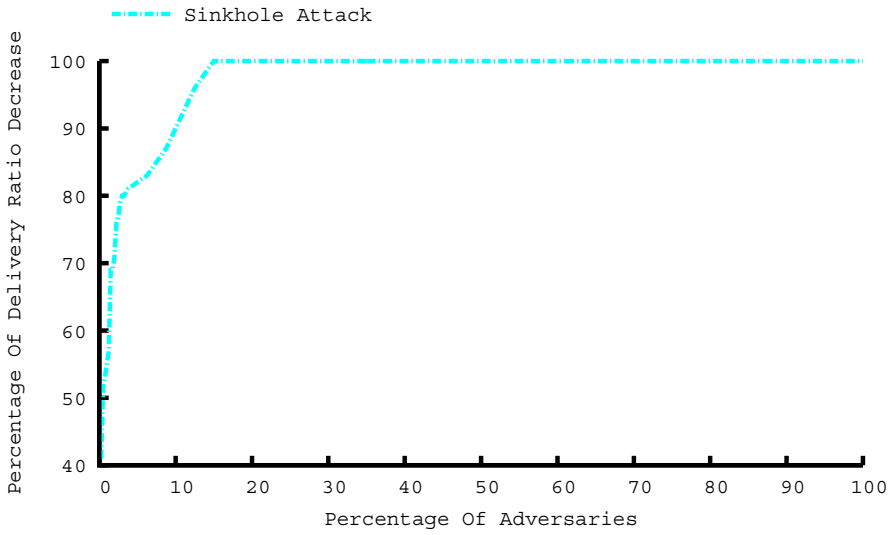

Figure.15. Delivery ratio reduction under Sinkhole attack.

In order to apply a hybrid attack, we combined the one-hop routing misdirection to Sinkhole attack. As three adversaries are sufficient to half the percentage of delivery ratio, we chose three adversaries to apply 1-hop route misdirection in addition to sinkhole attack. Figure.16. illustrates the results. Again, we chose the adversaries near the target region to be more effective. As can be seen the delivery ratio is decreased significantly. This is because the adversaries causes a significant number of messages do not reach the destination and get lost in the network.

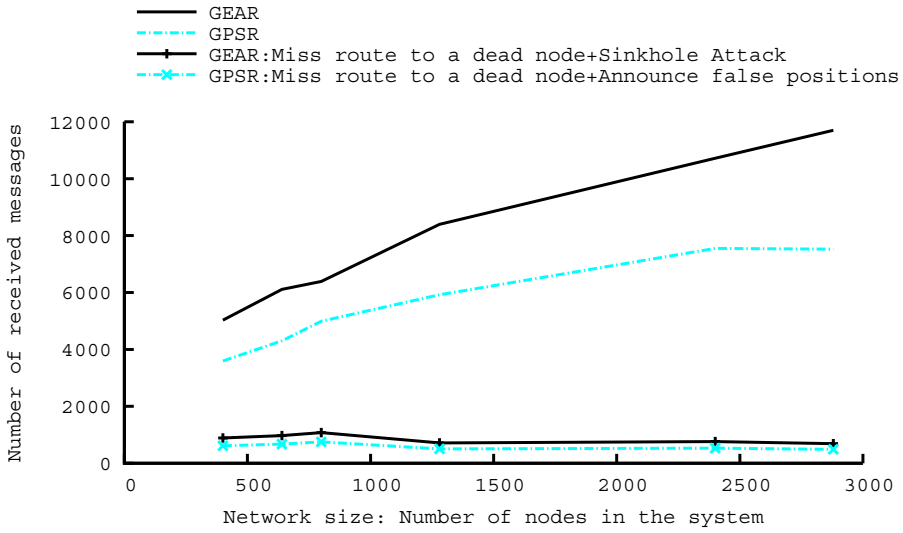

Figure.16. # of received messages under Routing misdirection and Sinkhole attack.

One of our mitigation techniques against data forwarding phase attacks is disjoint multipath routing. However, before applying this technique on a network under attack, we need to measure to what extent this approach affects the efficiency of a network in normal situation when there is no malicious node in the network.
Figure.17. shows the number of delivered messages after applying the disjoint multipath on a network without any malicious node. As shown, GPSR shows a better performance than GEAR. The reason is that GEAR is not a loop free protocol by nature and generates routing loops. There come situations when more than one of the neighbors are equally the best choice. For example, Figure.17. shows when number of nodes are 1400; we see a considerable decrease, which is caused by routing loops. The approach works better for GPSR, as it is a loop free protocol.

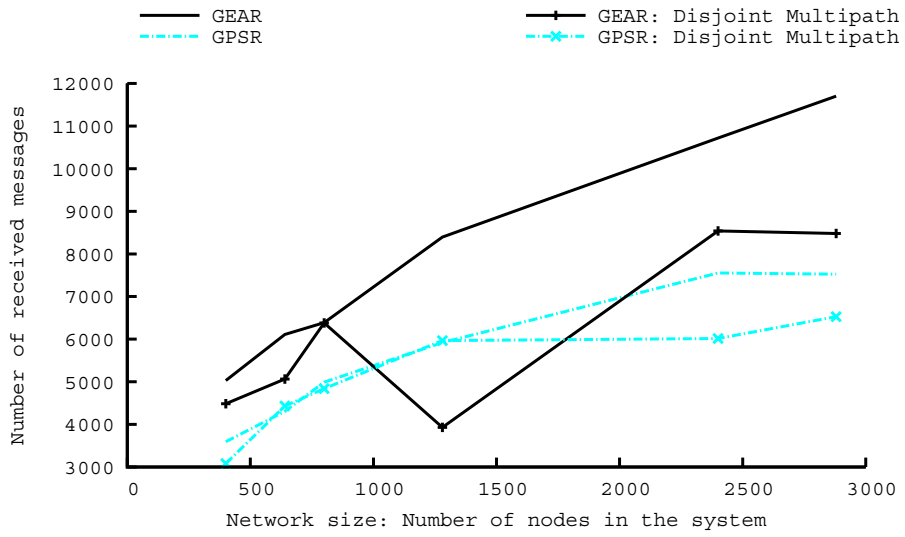

Figure.17. # of received messages in multipath routing before attack.

After applying multipath routing as the mere mitigation approach, the performance showed a slight delivery ration increase. As described in detail in section 4.3.2, this method alone is not sufficient to increase the delivery ratio and should be combined with statistical analysis of the system history. Figure.18. shows the result of applying the hybrid technique on GEAR and Figure.19. shows the result for GPSR. As can be seen, the use of this technique nearly doubled the number of delivered messages even under both routing misdirection and Sinkhole attack.

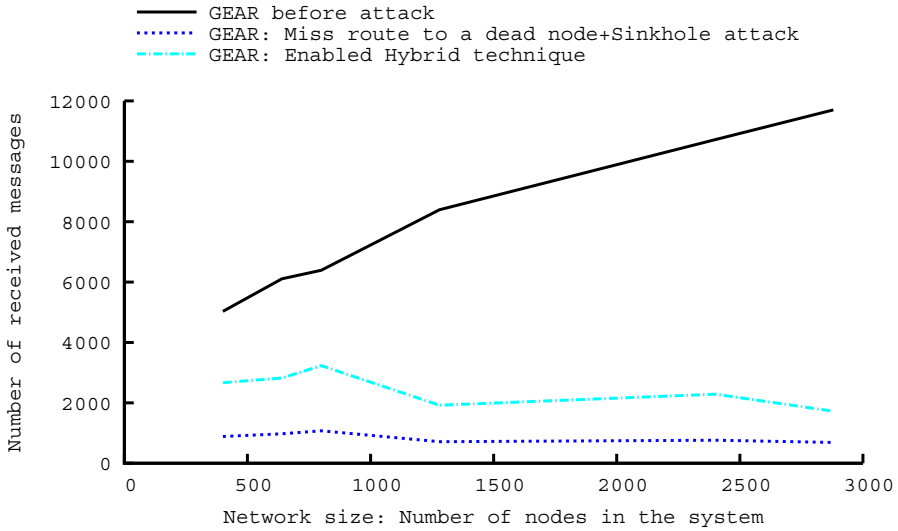

Figure.18. # of received messages in GEAR after applying the Hybrid technique.

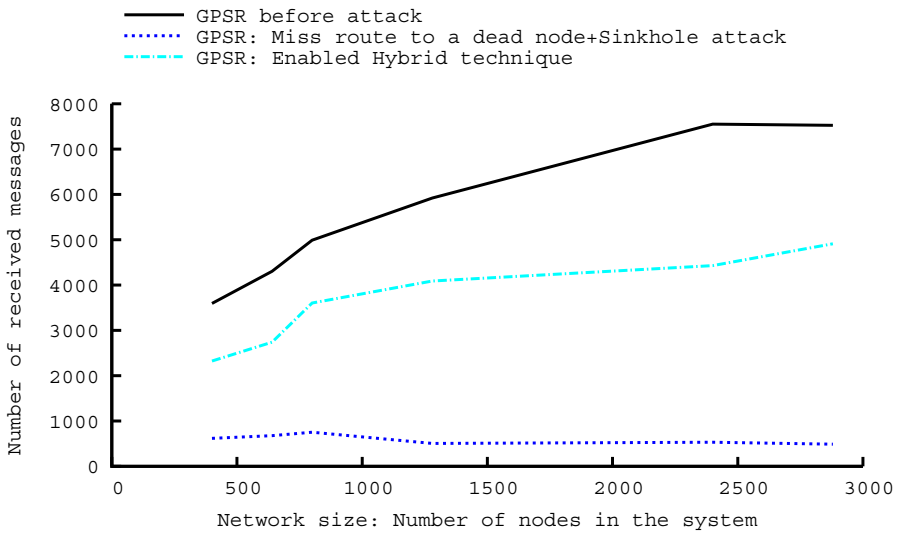

Figure.19. # of received messages in GPSR after applying the Hybrid technique.

6.4 Byzantine Attack

In this attack the magnitude of the covered area by adversaries is more important the number of them. In order to form routing loops to apply the Byzantine attack, we tried adversaries located on circles with different radii. The adversaries need to be in the radio range of each other. Meanwhile the loop should not be very wide. Since the adversaries are mote class adversaries, to make sure that their intended attack is successful they need to be close to each other. As we described in section 4.3.3, if adversaries are too far apart, we need laptop class adversaries and the attack will change to wormhole attack. Since the maximum radio range is 25 in our simulation model, the adversaries can form a circle with radius of up to 5. We also tried loops in circles with radius 4, 3, 2 and 1. Number of adversaries forming a circle of radius 5, 4 and 3 is 12. For radius 2, it is 8 and for a circle of radius 1 we have 4 adversaries. Figure.20. shows the reduction of delivery ratio based on the radius of the routing loop. As can be seen a loop with radius 5 reduces the delivery ratio about 40%.

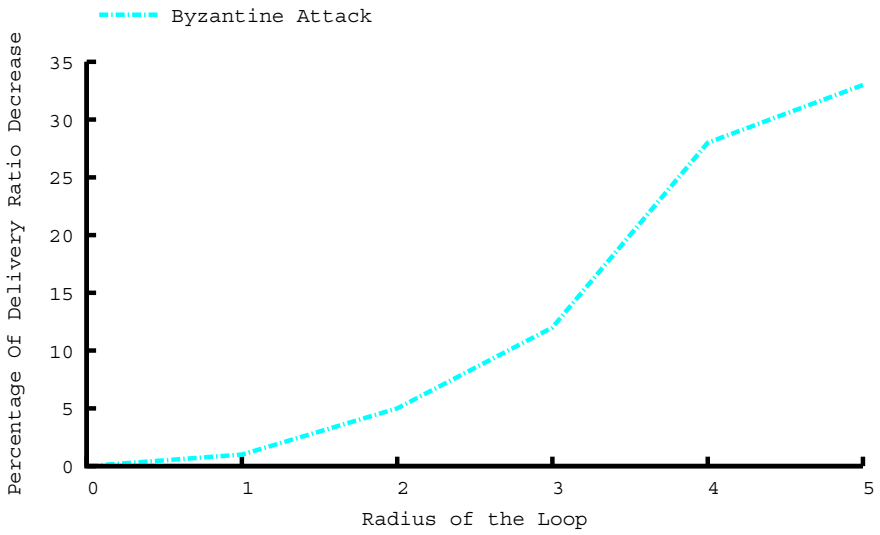

Figure.20. Delivery ratio reduction under Byzantine attack.

We choose the loop with radius 4, the adversaries are 12 nodes located on the loop. Upon receiving the packet, they forward a random percentage of received packets in a legitimate way. For the rest of packets, they keep on forwarding the packet to one another and form a loop that leads to energy consumption and packet loss. The result of this attack is shown in Figure.21.

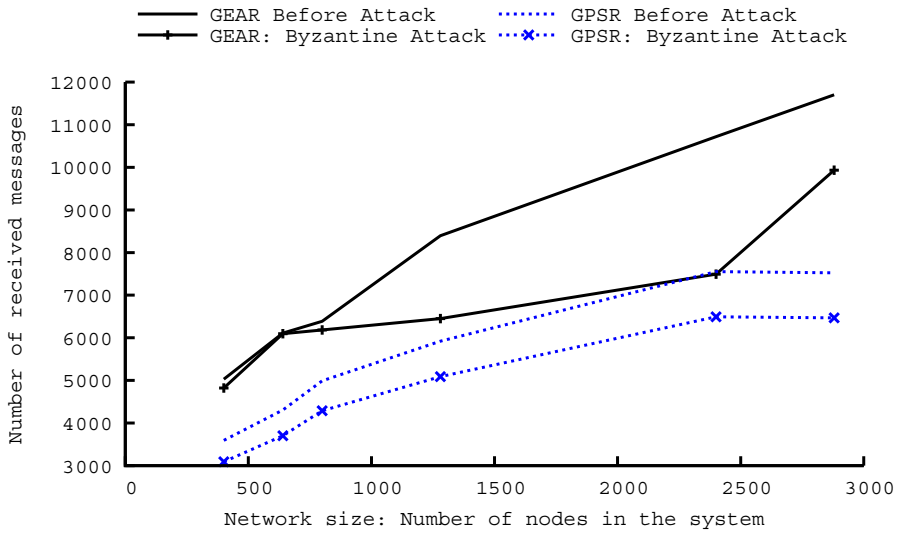

Figure.21. # of received messages in GEAR and GPSR under Byzantine attack.

As mentioned, each node needs to maintain the <source, sequence number> of the last 10 received messages. The length of the partial trace in message header is 5, as Figure.22. shows, we tried length of 2, 3, 4 and the generated loops by GEAR just got bigger yet not broken. When the length reached to 5 almost 90 percent of packets were rescued from the GEAR loops and remained fixed with larger values.

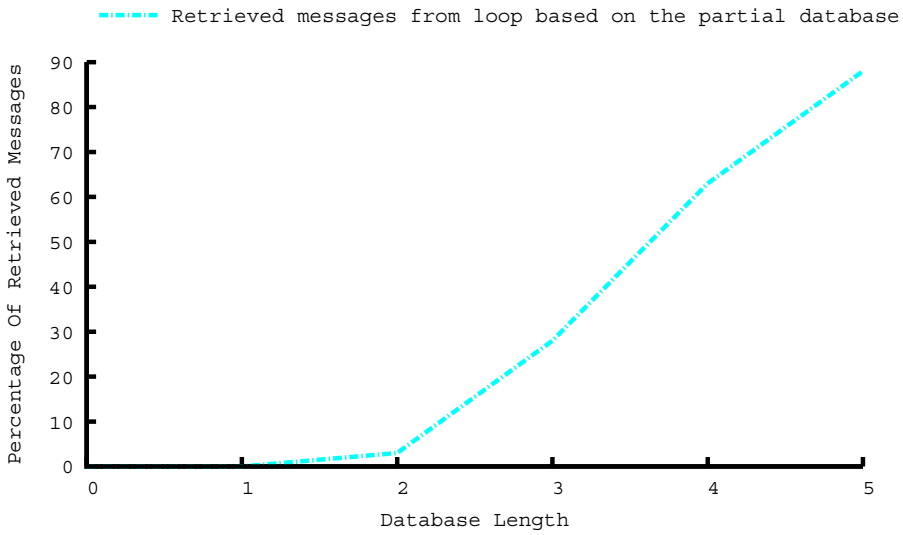

Figure.22. Percentage of retrieved messages from loop, based on the database length.

The initial neutral trust value is set to 1. First, we chose the additive decrease method for trust value with every misbehavior detection, which did not show much difference. On the second try, the trust value is halved and in addition, forerunner messages are sent to notify other

nodes about the malicious nodes. This time, we observed a quite good difference. However since they are other messages in the queue in front of forerunner messages, other nodes are notified about the id of adversaries with delay. Therefore, on the third try, with the aim of speeding up the receiving of forerunner messages, the message queue is sorted and the forerunner messages are pushed to the head of the queue. The results are shown in Figure.23. for GPSR and Figure.24. for GEAR. Again, GPSR shows a better performance. Because there were situations, in which Forerunner messages and update message conflict with each other and the message queue was overflowed.

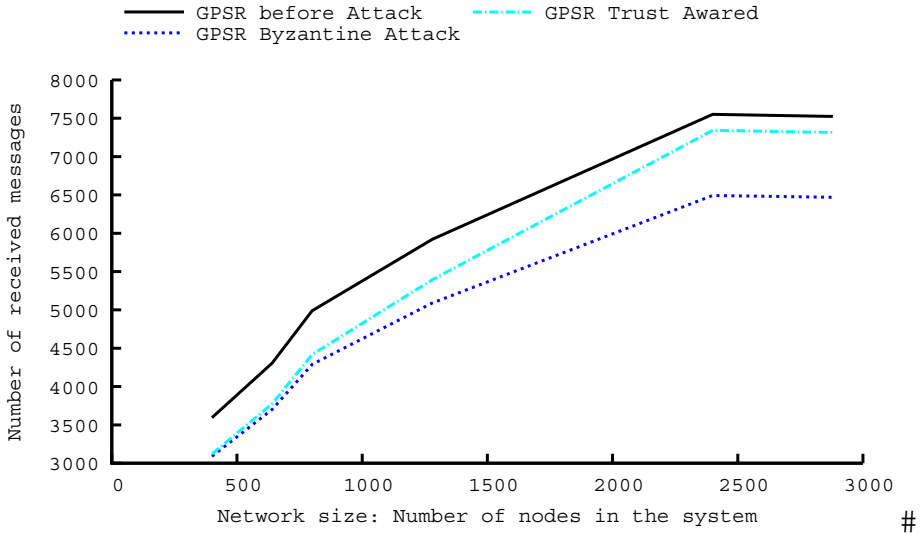

Figure.23. # of received messages after making GPSR Trust-Aware.

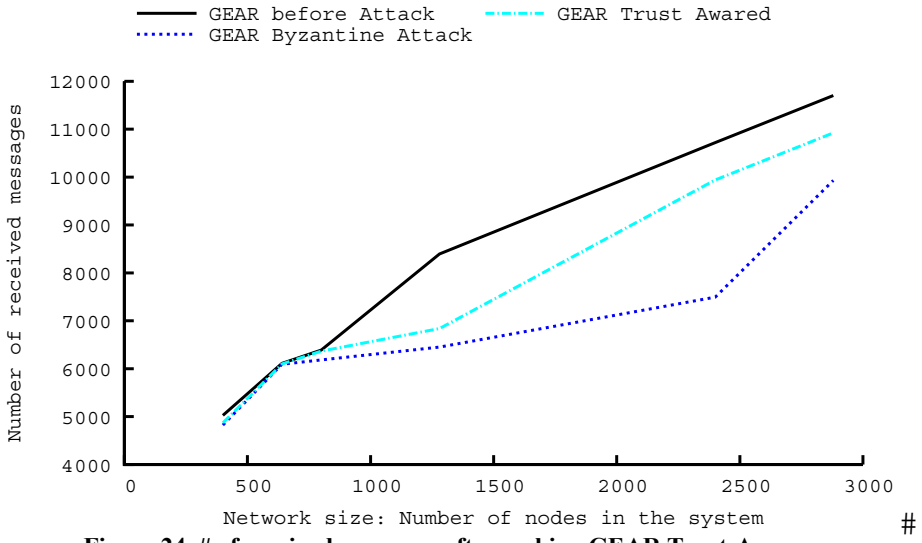

Figure.24. # of received messages after making GEAR Trust-Aware.

7. Conclusion and Future Work

In this work, we studied some of the attacks against GEAR and GPSR routing protocols. The attacks targeted the reliance of these protocols on confidentiality, integrity and availability. We classified these attacks as data forwarding attacks, path selection attacks and impersonation attacks. As hybrid attacks are more effective and harder to detect we put GEAR and GPSR under hybrid attacks, from the first two categories we chose, selective forwarding, routing misdirection, sinkhole attack and byzantine attack, from the third category we selected the Sybil attack. We then demonstrated the impact of the attacks on the performance and network lifetime and saw that although each of these attacks is mounted by mote class adversaries, they can cause considerable harm to the system performance. We showed that sinkhole attack could be one of the most effective attacks especially if the adversary is near the destination of messages and even a small percentage of the adversaries can decrease the delivery ratio significantly. We presented that if routing misdirection is applied only to one hop, it will not be very efficient and a group of colluding adversaries is needed to make this attack efficient. This is called the byzantine attack, which falls under the category of routing misdirection attacks. We exhibited that in this attack, the magnitude of the area under attack is more important than the number of adversaries. We then developed various detection and mitigation techniques in order to avoid these attacks. Our proposed methods required no specialized hardware and did not rely on the cryptographic schemes. Therefore, they were efficient in terms of all three dimensions of computation, storage and bandwidth that has made them suitable for resource constrained WSNs. We needed to pay the price of a hardware independent and cryptography free solution by developing pure software solutions. One of the proposed solutions was to use an alternative path parallel to the primary path along with trust based path selection in order to avoid mistrusted routes dynamically and to continue routing in the presence of attacks. Another solution was keeping a partial history of the past messages in every node along with a partial history of the visited nodes in every sent message. Although the required memory to store the history is not very compatible with the basic principle of geographic routing which underlines that every node should keep o(d) state and no message should carry more than O(log n) control bits for n nodes, it is still far less than the overhead caused by conventional source-routed, Distance Vector, and Link State algorithms. Therefore, geography is still a powerful lever to scale routing. By the way, tests have shown that, it is the number of exchanged messages that play a decisive role in the performance and network lifetime, not the trivial additional space we added in the nodes and exchanged messages. To take arms against Sybil attack we used a protection scheme, which had no requirement for secret information, shared keys or special hardware support. We then demonstrated the impact of the effectiveness and efficiency of our mitigation techniques using a hybrid application implemented for both GPSR and GEAR in the TOSSIM simulator. The experiment results show that both GEAR and GPSR show the same behavior under attacks and the efficiency of both protocols is reduced to the same degree. However, in case of defense techniques GPSR shows a far better performance than GEAR. In fact, two of our mitigation techniques did not work very efficient for GEAR because of GEAR frequent update messages. These update messages are co-overload the network with attack detector broadcasts and notification broadcasts after discovering the adversaries.

Nevertheless, this is not the case with GPSR since nodes receive update information through piggy backed content messages and do not need separate frequent update messages. As a result, all of our proposed mitigation techniques worked with GPSR and improved the performance reasonably.

In addition to all the discussed problems, some open issues are remained that are worthy of a complementary research. We handled the trust value adjustment in a simple and elementary way. Typically, trust value adjustment is a critical and delicate matter. The system efficiency could be improved even more in case of further investigation and empirical tests on trust value adjustment. All scenarios were tested for grid networks. Some mitigation techniques such as neighbor analysis approach might generate false alarms in sparse and random graph networks. In addition, GPSR always works better than GEAR in sparse networks. Thus, results could be different for both GPSR and GEAR if the network topology was a random graph. Finally, we did not study the security of the Face Routing component. With the assumption that the algorithm of the component is complex, we then assumed that the adversary has less incentive to attack it. However, the door is open and the adversary can attack this component with more efforts. To the best of our knowledge, no research has been done in this context up to this moment.

8. Appendix

Abbreviation	Description
AES	Advanced Encryption Standard
AODV	Ad hoc On-Demand Distance Vector Routing
CBC–CS	Cipher Block Chaining Cipher Block Chaining–Cipher text Stealing
CBC–MAC	Cipher Block Chaining Message Authentication Code
CIA	Confidentiality, Integrity, Availability
CMAC	Cipher-based Message Authentication Code
CPM	Closet Pattern Matching
CRUD	Create, read, update, delete
DES	Data Encryption Standard
GAF	Geographic Adaptive Fidelity
GEAR	Geographic Energy Aware Routing
GFG	Greedy-Face-Greedy
GHT	Geographic Hash Table
GOAFR	Greedy Other Adaptive Face Routing
GPS	Global Positioning System
GPSR	Greedy Perimeter Stateless Routing
MAC	Message Authentication Code/ Media Access Control
NMS	Neighborhood Monitoring System
OCB mode	Offset Codebook Mode
OLSR	Optimized Link State Routing
RC5	Rivest Cipher 5
SEAR	Secure Energy Aware Routing
SNEP	Secure Network Encryption Protocol
SPINS	Security Protocols for Sensor Networks
TARF	Trust Aware Routing Framework
TESLA	Timed Efficient Stream Loss-Tolerant Authentication
TKIP	Temporal Key Integrity Protocol
TRANS	Trust Routing for Location Aware Sensor Networks
WSN	Wireless Sensor Networks
XTEA	eXtended Tiny Encryption Algorithm

Table.7. List of abbreviations

9. References

[1] Dargie, W. and Poellabauer, C. Fundamentals of wireless sensor networks: theory and practice. *John Wiley and Sons. ISBN 978-0-470-99765*, 2010.

[2] Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Published in Magazine communications of the ACM Volume 17 Issue 11, Pages 643 – 644,* 1974.

[3] Jaap-Henk Hoepman, Andreas Larsson, Elad M. Schiller and Philippas Tsigas. Secure and self-stabilizing clock synchronization in sensor networks. *Published in: Theoretical Computer Science, vol. 412, no. 40. Pages 5631–5647,* 2011.

[4] Jaap-Henk Hoepman, Andreas Larsson, Elad M. Schiller and Philippas Tsigas. Secure and self-stabilizing clock synchronization in sensor networks. *In Proc. of the 9th International Symposium on Self Stabilization, Safety, And Security of Distributed Systems,* 2007.

[5] Aaron Zollinger. Networking Unleashed: Geographic Routing and Topology Control in Ad Hoc and Sensor Networks. A dissertation submitted to the Swiss Federal Institute of Technology Zurich for the degree of Doctor of Sciences, 2005.

[6] Ke Liu. Towards Practical and Resilient Geometric Routing for Wireless Sensor Networks. Ph.D. Prospectus in the Graduate School of Binghamton University State University of New York, 2008.

[7] Young Jin Kim, Ramesh Govindan, Brad Karp and Scott Shenker. On the Pitfalls of Geographic Face Routing. *In Proc. Of the 2005 joint workshop on Foundations of mobile computing Pages 34 – 43,* 2005.

[8] Brad Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. *In Proc. of the 6th annual international conference on Mobile computing and networking. Pages 243-254*, 2000.

[9] Yan Yu, Ramesh Govindan and Deborah Estrin. Geographical and Energy Aware Routing: a recursive data dissemination protocol for wireless sensor networks, 2001.

[10] M. Al-Shurman, S.-M. Yoo, and S. Park, Black hole attack in mobile ad hoc networks. *In Proc. of the 42nd annual Southeast regional conference Pages 96-97,* 2004.

[11] B. Xiao, B. Yu, and C. Gao. Detection and localization of Sybil nodes in VANETs. *In Proc. of the 2006 Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks,* 2006.

[12] H. Yu, M. Kaminsky, P. B. Gibbons and A. Flaxman. Sybil Guard defending against Sybil attacks via social networks. *In Proc. of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications Pages 267-278,* 2006.

[13] R. Muraleedharan, Y. Yan and L. A. Osadciw. Detecting Sybil attacks in image sensor network using cognitive intelligence. *In Proc. of the First ACM Workshop on Sensor and Actor Networks,* 2007.

[14] A. A. Pirzada and C. McDonald. Establishing trust in pure ad-hoc networks. *In Proc. of the 27th Australasian Conference on Computer Science, Vol. 26, pages 47–54,* 2004.

[15] S. Marti, T. J. Giuli, K. Lai and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. *In Proc. of the 6th Annual International Conference on Mobile Computing and Networking, pages 255–265,* 2000.

[16] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *In Proc. of IEEE International Workshop on Ad Hoc Networks,* 2003.

[17] James Newsome, Elaine Shi, Dawn Song and Adrian Perrig. The Sybil Attack in Sensor Networks: Analysis & Defenses. *In Proc. of the 3rd international symposium on Information processing in sensor network Pages 259-268,* 2004.

[18] Kuo-Feng Ssu, Wei-Tong Wang and Wen-Chung Chang. Detecting Sybil attacks in Wireless Sensor Networks using neighboring information. *Published in Journal Computer*

Networks: The International Journal of Computer and Telecommunications Networking archive Volume 53. Pages 3042-3056, 2009.

[19] B. Awerbuch, D. Holmer, C. Nita-Rotaru and H. Rubens. An on-demand secure routing protocol resilient for byzantine failures. *In Proc. of ACM Workshop on Wireless Security, Atlanta, GA*, 2002.

[20] Claude Cr´epeau, Carlton R. Davis and Muthucumaru Maheswaran. A secure MANET routing protocol with resilience against byzantine behaviors of malicious or selfish nodes. *In Proc. of the 21th International Conference on Advanced Information Networking and Applications Workshops Volume 02 Pages 19-26*, 2007.

[21] Jaydip Sen. A Survey on Reputation and Trust-Based Systems for Wireless Communication Networks, 2011.

[22] Gay, D., Levis, P., and Culler, D. 2007. Software design patterns for TinyOS. *Published in Journal ACM Transactions on Embedded Computing Systems (TECS), Volume.6*, 2007.

[23] Deepak Ganesan, Ramesh Govindan, Scott Shenker, Deborah Estrin. Highly-Resilient, Energy-Efcient Multipath Routing in Wireless Sensor Networks. *Published in: Newsletter ACM SIGMOBILE Mobile Computing and Communications,Volume 5. Pages 11-25*, 2001.

[24] J. S´a Silva, B. Krishnamachari, and F. Boavida. TARF: A Trust-Aware Routing Framework for Wireless Sensor Networks. *In Proc. GLOBECOM Workshops (GC Wkshps), 2010.*

[25] Honglong Chen and Wei Lou and Zhi Wang. A Novel Secure Localization Approach in Wireless Sensor Networks. *Published in: Journal EURASIP Journal on Wireless Communications and Networking Special issue on design, implementation, and evaluation of wireless sensor network systems. Volume 2010 Article No. 12,* 2010.

[26] Philip Levis, TinyOS Programming June 28, 2006.

[27] Tanya Roosta, Shiuhpyng Shieh and Shankar Sastry. Secure routing in wireless sensor networks: attacks and countermeasures. *In Proc. of the First IEEE. 2003 IEEE International Workshop,* 2003.

[28] Jing Dong, Kurt E. Ackermann, Brett Bavar and Cristina Nita Rotaru. Secure and Robust Virtual Coordinate System in Wireless Sensor Networks. *Published in: Journal ACM Transactions on Sensor Networks, Vol. 6, No. 4, Article 29,* 2010.

[29] Mohammed Erritali, Oussama Mohamed Reda and Bouabid El Ouahidi. A contribution to secure the routing protocol: Greedy perimeter stateless routing using a symmetric signature-based AES and MD5 hash. In *Proc. of the 11th WSEAS international conference on applied computer science,* 2011.

[30] Donggang Liu and Peng Ning, Multi-Level μTESLA: Broadcast Authentication for Distributed Sensor Networks. *Published in: Journal ACM Transactions on Embedded Computing Systems (TECS) TECS, Volume 3 Pages 800 – 836,* 2004.

[31] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. *Published in: Journal Wireless Networks Volume 8, Pages 521- 534,* 2002.

[32] Chris Karlof, Naveen Sastry, David Wagner. TinySec: a link layer security architecture for wireless sensor networks. *In Proc. of the 2nd international conference on Embedded networked sensor systems Pages 162-175*, 2004.

[33] Dunkels A., Grönvall B. and Voigt T. Contiki a Lightweight and Flexible Operating System for Tiny Networked Sensors. *In Proc. of the 29th Annual IEEE International Conference on Local Computer Networks, Pages 455—462,* 2004.

[34] Lander Casado and Philippas Tsigas. ContikiSec: A Secure Network Layer for Wireless Sensor Networks under the Contiki Operating System. *In Proc. of the 14th Nordic*

Conference on Secure IT Systems: Identity and Privacy in the Internet Age Pages 133 – 147, 2009.

[35] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Published in Journal: Computer Networks: The International Journal of Computer and Telecommunications Networking archive Volume,* 2007.

[36] Di Tang, Tingting Jiang and Jian Ren. Secure and Energy Aware Routing (SEAR) in Wireless Sensor Networks. *In Proc. Global Telecommunications Conference (GLOBECOM 2010),* 2010.

[37] Sapon Tanachaiwiwat, Pinalkuniar Davel, Rohan Bhindwale and Ahmed Helmyl. Location-centric isolation of misbehavior and trust routing in energy-constrained sensor networks. *In Proc. Performance, Computing, and Communications, 2004 IEEE International Conference,* 2004.

[38] Sen, J. and Ukil, A. A secure routing protocol for wireless sensor networks. *In Proc. of the International Conference on Computational Sciences and its Applications (ICCSA'10), Pages 277 – 290*, 2010.

[39] Philip Levis and Nelson Lee. TOSSIM: A Simulator for TinyOS Networks, 2003.

[40] Hyung June Lee, Alberto Cerpa and Philip Levis. Improving wireless simulation through noise modeling. *In Proc. the 6th international conference on Information processing in sensor networks Pages 21-30,* 2007.