2009

Trusted Querying over Wireless Sensor Networks and Network Security Visualization

Giovani Rimon Abuaitah
Wright State University

Follow this and additional works at: https://corescholar.libraries.wright.edu/etd_all

Part of the Computer Engineering Commons

Repository Citation
https://corescholar.libraries.wright.edu/etd_all/273

This Thesis is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.
TRUSTED QUERYING OVER WIRELESS SENSOR NETWORKS AND
NETWORK SECURITY VISUALIZATION

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

By

GIOVANI RIMON ABUAITAH
B.S., Birzeit University, 2006

2009
Wright State University
I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Giovani Rimon Abuaitah ENTITLED Trusted Querying over Wireless Sensor Networks and Network Security Visualization BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science

______________________________________________

Bin Wang, Ph.D.
Thesis Director

______________________________________________

Thomas Sudkamp, Ph.D.
Department Chair

Committee on Final Examination

______________________________________________

Bin Wang, Ph.D.

______________________________________________

Yong Pei, Ph.D.

______________________________________________

Thomas Wischgoll, Ph.D.

______________________________________________

Joseph F. Thomas, Jr., Ph.D.
Dean, School of Graduate Studies
ABSTRACT


Wireless sensor networks (WSNs) as an emerging technology faces numerous challenges. Sensor nodes are usually resource constrained. Sensor nodes are also vulnerable to physical attacks or node compromises. Answering queries over data is one of the basic functionalities of WSNs. Both resource constraints and security issues make designing mechanisms for data aggregation particularly challenging. In this thesis, we first explore the various security techniques for data aggregation in WSNs then we design and demonstrate the feasibility of an innovative reputation-based framework rooted in rigorous statistical theory and belief theory to characterize the trustworthiness of individual nodes and data queries in WSNs.

Detecting security vulnerabilities is an imperative task. Visualization techniques have been developed over decades and are powerful when employed in the field of network security. In this thesis, we present a novel security visualization tool called “SecVizer”.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF FIGURES</th>
<th>viii</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>x</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>xi</td>
</tr>
</tbody>
</table>

## Chapter

### I. INTRODUCTION

1. Wireless Sensor Networks
    1.1 Spatio-Temporal Correlation
    1.2 Network Lifetime
    1.3 Design Characteristics
    1.4 Security
    1.5 Trusted Querying

2. Network Security Visualization

3. Thesis Contribution and Outline

### II. IN-NETWORK DATA AGGREGATION

1. Overview

2. Aggregation Schemes
   2.1 TAG
   2.2 LEACH
   2.3 Synopsis Diffusion
   2.4 Tributaries and Deltas
   2.5 CountTorrent
2.6 Approximate Aggregation Techniques ................................................................. 18

3. Secure Data Aggregation ............................................................................................ 19

III. TRUST MANAGEMENT IN SENSOR NETWORKS ......................................................... 21

1. Reputation and Trust Definition .................................................................................. 21

2. Trust Establishment ..................................................................................................... 22

3. Attacks on Reputation and Trust-Based Schemes ....................................................... 24

   3.1 Bad Mouthing Attack .............................................................................................. 25

   3.2 On-off Attack .......................................................................................................... 25

   3.3 Conflicting Behavior Attack .................................................................................. 26

   3.4 Sybil Attack and Newcomer Attack ....................................................................... 26

IV. TRUSTED QUERY IN SENSOR NETWORKS ................................................................ 27

1. Introduction .................................................................................................................. 27

2. Reputation-based Spatial Temporal Correlated Sensing Framework ......................... 29

3. Sensor Node Reputation Characterization and Update ............................................. 33

   3.1 Relative entropy based scheme ............................................................................. 34

   3.2 Consistency based scheme .................................................................................... 35

4. Sensor Node Classification and Compromised Node Detection ............................... 37

5. Aggregation Result Uncertainty Quantification ......................................................... 39

6. Simulation Evaluation .................................................................................................. 43

   6.1 Sensor Node Reputation Evolution ........................................................................ 43

   6.2 Aggregation Result and Belief of Result with Misbehaving Nodes ....................... 45

   6.3 Impact of Cooperative Malicious Node .................................................................. 47

7. Summary ...................................................................................................................... 48
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Typical WSN Architecture</td>
<td>2</td>
</tr>
<tr>
<td>2.</td>
<td>A Web-based Sensor Networks Monitoring System</td>
<td>10</td>
</tr>
<tr>
<td>3.</td>
<td>Efficiency of In-Network Data Aggregation</td>
<td>13</td>
</tr>
<tr>
<td>4.</td>
<td>LEACH Clustering-based Scheme</td>
<td>16</td>
</tr>
<tr>
<td>5.</td>
<td>Synopsis Diffusion Multi-path Scheme</td>
<td>17</td>
</tr>
<tr>
<td>6.</td>
<td>Trust Constructs in Computer Networks</td>
<td>22</td>
</tr>
<tr>
<td>7.</td>
<td>Trust Propagation for Indirect Trust Establishment</td>
<td>23</td>
</tr>
<tr>
<td>8.</td>
<td>A Schematic Illustration of a Reputation-based Spatial Temporal Correlated Sensing Framework.</td>
<td>30</td>
</tr>
<tr>
<td>10.</td>
<td>An Example where the Opinion about a Proposition ( x ) from a Binary State Space Has the Value ( \omega_x = (0.7,0.1,0.2,0.5) ) [20]</td>
<td>40</td>
</tr>
<tr>
<td>11.</td>
<td>Drive Trust from Parallel Transitive Paths</td>
<td>42</td>
</tr>
<tr>
<td>12.</td>
<td>An Example Logical Hierarchical Topology Used in QualNet Simulation</td>
<td>43</td>
</tr>
<tr>
<td>14.</td>
<td>A Snapshot of Reputation of Sensor Nodes</td>
<td>45</td>
</tr>
<tr>
<td>15.</td>
<td>Aggregate Sensor Readings at an Aggregator</td>
<td>45</td>
</tr>
<tr>
<td>16.</td>
<td>Expected Belief Value at the Aggregator That Measures the Uncertainty in the Aggregate Sensor Reading</td>
<td>46</td>
</tr>
<tr>
<td>17.</td>
<td>Aggregate Sensor Readings at the Cluster Head</td>
<td>46</td>
</tr>
<tr>
<td>18.</td>
<td>Expected Belief Value at the Cluster Head That Measures the Uncertainty in the Query Response</td>
<td>47</td>
</tr>
<tr>
<td>20.</td>
<td>iNSpect Simulation Visualization</td>
<td>51</td>
</tr>
<tr>
<td>21.</td>
<td>rumint Parallel Coordinate Plot</td>
<td>52</td>
</tr>
<tr>
<td>22.</td>
<td>SecVizer System Architecture</td>
<td>53</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>23. QualNet Nodes Positioning File Structure (.nodes)</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>24. QualNet Traffic Trace File (.trace) Format</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>25. Flowchart of the Process of Parsing the QualNet Trace File</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>26. SecVizer Graphical User Interface (GUI)</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>27. SecVizer Parallel Coordinate Plot Axes Support</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>28. SecVizer Nodes Statistics Window</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>29. QualNet Topology Layout</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>30. SecVizer Parallel Coordinate Plots of Different Simulated Security Scenarios</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>31. SecVizer Topology Window Snapshots of Different Security Scenarios</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>32. Nodes Statistics for Port Scan</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>33. Nodes Statistics for Host Scan</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>34. SecVizer Sequence Diagram Illustrating the Interactions among the Different Windows</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>35. SecVizer Class Diagram</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>36. Load Topology Slot Code Implementation</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>37. The Main OpenGL Drawing Function under the Topology Window</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>38. Code Implementation of the Topology Rendering Function</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>39. Code Implementation of the Active Records Rendering Function</td>
<td>78</td>
<td></td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Characteristics of Sensor Nodes</td>
<td>4</td>
</tr>
<tr>
<td>2. Open Source Network Visualization Tools</td>
<td>9</td>
</tr>
<tr>
<td>3. Data Aggregation Schemes</td>
<td>18</td>
</tr>
<tr>
<td>4. Description of the First Line Fields of Figure 23</td>
<td>54</td>
</tr>
<tr>
<td>5. Description of Figure 24 Trace Record</td>
<td>55</td>
</tr>
<tr>
<td>6. Action Code Map</td>
<td>55</td>
</tr>
<tr>
<td>7. Summary of SeeVizer Required Libraries</td>
<td>70</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

First of all, I would like to thank my advisor Dr. Bin Wang for his tremendous help and support throughout my stay at the Broadband, Mobile and Wireless Networking Research Laboratory at Wright State University and for the persisting positive feedbacks that definitely helped me complete this thesis. Without his help, this task could have never been accomplished. I would also like to thank Dr. Yong Pei for his continuous encouragement in researching into the field of sensor networks. Special thanks go to Dr. Thomas Wischgoll for his help in technical issues regarding visualization and for his constant presence when complications arise.

My extreme gratitude goes to my brother Wadie for his support during school stay at Dayton, my brothers Rami and Marco for their love and support and at last but not least, my father Rimon and my mother Linda for their enduring encouragement in pursuing my graduate studies.

Finally, I would like to take the opportunity to thank my fabulous laboratory colleagues and my close friends that were always there when stress begins. They were my family whenever my closest relatives were not around.
To my beloved parents and my dear brothers
I. INTRODUCTION

The advancements in micro electronics and wireless communications have led to the creation of the wireless sensor network (WSN) technology. This technology has many applications, including various environmental monitoring. A primitive objective of WSNs is to answer queries by gathering sensory data from the deployed sensors; the process of collecting sensory data is often called “in-network processing” or “aggregation”. Since sensor nodes in WSN technology are usually tiny micro-electronic devices which have limited resources (low processor speed, small memory size, low computation and communication power), it becomes very challenging to design mechanisms to support data queries. On the other hand, the monitoring environments, where the sensor network technology is being employed, are usually hostile in nature and are vulnerable to physical tampering where an attacker can compromise the sensor node and launch hazardous attacks from there. This security vulnerability adds a new challenge to the design of secure mechanisms for sensor networks. Detecting such vulnerabilities is considered a crucial task. Various techniques have been developed and studied, including network security visualization techniques.

In this chapter, we give an introduction to wireless sensor networks and network security visualization. Section 1 discusses a common characteristic in WSNs called “spatio-temporal correlation”, defines an important concept in WSNs called the “network lifetime”, overviews the design characteristics of such networks, discusses the security issues in sensor networks and at the end provides an overview of the essential needs for the trusted querying approach. Section 2 addresses visualization in network security. We summarize the thesis contributions in Section 3. Data aggregation and its relevant security mechanisms are discussed separately in Chapter II whereas details of trust management in sensor networks are provided in Chapter III.
1. Wireless Sensor Networks

Wireless sensor networks (WSNs) have recently emerged as a technology that has resulted in a variety of applications. Many applications such as health care, medical diagnostics, disaster management, military surveillance, and emergency response have been deploying such networks as their main monitoring framework [1]. Basically, a wireless sensor network consists of a number of tiny sensor nodes connected together through wireless links. Some more powerful nodes may operate as control nodes called base stations. Often, the sensing nodes are referred to as “motes” while base stations are sometimes called “sinks”. Each sensor node can sense data from its surroundings (e.g. temperature, humidity, pressure), conduct simple computations on the collected data and send it to other neighboring nodes through the communication links. Control nodes may further process the data and probably transfer it to a database server via a wired connection. Figure 1 shows a typical architecture for a WSN. The sensing nodes “motes” are represented by black spheres and are responsible for observing the surrounding environment whereas the cube represents a control node “sink” which serves as the base station.

Figure 1. Typical WSN Architecture
1.1 Spatio-Temporal Correlation

Correlation among the sensor observations is a unique and significant characteristic of WSNs, a characteristic that can be exploited to drastically enhance the overall network performance [8] [9]. Two common correlation characteristics are realized in properly deployed sensor networks:

1) Spatial Correlation: Usually, sensors in WSNs are densely populated over a region. Spatial proximity of sensors, therefore, makes the region observations highly correlated. The degree of correlation may further increase by the decrease of inter-node separation.

2) Temporal Correlation: Typically, sensor nodes periodically report their observations of a specific phenomenon. The temporal correlation degree between any consecutive sensor readings may vary depending on the nature of the physical phenomenon.

It is to be noticed that throughout the discussion of this thesis, we usually assume that all deployed sensors are spatially and temporally correlated, meaning that they are geographically close to each others and report measurements of the environment almost at the same time. Therefore, correlated sensors share similarities in their observations of the surroundings (e.g., close temperature readings).

1.2 Network Lifetime

Network lifetime is a very important concept in WSNs. Typically, applications involving WSNs require the whole network to operate at least for a given mission time or as long as possible; this is what is known as the network lifetime [7]. Network lifetime can be defined as the time for which the network is operational or the time during which the network is able to fulfill its tasks starting from a given amount of stored energy. Because wireless sensor networks are resource constrained: limited power supply, bandwidth for communication, processing speed, and memory, the objective therefore is to reduce the energy consumed by the sensor nodes and
thus maximize the lifetime of the network. How to achieve this? We may apply lightweight mechanisms which reduce the amount of energy consumed by the sensors, and as a result maximize the run time of those sensors that keep the network alive.

1.3 Design Characteristics

Wireless sensors are designed to be tiny little devices with low cost. As examples, Table 1 shows the characteristics of SmartDust sensor nodes, MICAz motes [5], and SunSPOTs [6].

<table>
<thead>
<tr>
<th></th>
<th>SunSPOT (Sun)</th>
<th>MICAz (Crossbow)</th>
<th>SmartDust (Berkeley)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>32-bit 180MHz ARM920T core</td>
<td>8-bit 7.7MHz ATMega128</td>
<td>8-bit 4MHz</td>
</tr>
<tr>
<td>Storage</td>
<td>RAM 512K bytes</td>
<td>RAM 4K bytes</td>
<td>RAM 512 bytes</td>
</tr>
<tr>
<td></td>
<td>ROM 128K bytes</td>
<td>ROM 512 bytes</td>
<td>ROM 512 bytes</td>
</tr>
<tr>
<td></td>
<td>FLASH 4M byte</td>
<td>EEPROM 512K bytes</td>
<td>FLASH 8K bytes</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>250Kbps</td>
<td>250K baud</td>
<td>10Kbps</td>
</tr>
<tr>
<td>Default Payload Size</td>
<td>85 bytes</td>
<td>29 bytes</td>
<td>24 bytes</td>
</tr>
<tr>
<td>Operating System</td>
<td>Squawk VM (J2ME) No OS required</td>
<td>TinyOS</td>
<td>TinyOS</td>
</tr>
<tr>
<td>Power Supply</td>
<td>3.7V lithium-ion battery</td>
<td>2 AA batteries</td>
<td>2 AA batteries</td>
</tr>
</tbody>
</table>

Despite the noticeable difference among the three types of sensor nodes (the SunSPOT improves in the microprocessor speed, amount of storage, payload size and even in battery life), these devices are considered to be resource constrained. Clearly, when designing a mechanism for such devices, we have to take the following into consideration:

1) The low storage capability: The largest memory on board for the MICAz motes, for instance, can store up to 512Kbytes. A mechanism that stores a huge amount of data on the sensor nodes for future processing will not be efficient for such networks.
2) The low computational power: Energy resource of a sensor node is limited by size and cost constraints. For example, a MICAz mote will be deployed with non-rechargeable 2 AA batteries. Thus, we have to consider such limitation when designing a computational mechanism that utilizes the energy resource on the sensor nodes. A technique that consumes a significant amount of energy inhabited in the batteries during the computation process is not an energy efficient technique at all.

3) The communication overhead: Communicating wirelessly consumes more power at the nodes than any other activity, such as computation. Hence, it is crucial to design protocols so as to minimize the amount of communication required by the sensor nodes.

4) The unreliable wireless communication environment: Packet loss can happen due to packet errors or collision in WSNs. Since packet-based routing of the WSN is connectionless and wireless links in WSNs are bandwidth limited, a packet transmitted by one sensor may collide with another packet being sent by another sensor and consequently get dropped. Hence, as the probability of packet loss is high, we must design mechanisms that take this problem into account.

1.4 Security

Network security has become a very challenging topic especially when deploying the WSNs in a hostile environment. It is very important to provide such networks with the following security services [2]:

1) Authentication: There are two types of authentication in sensor networks; node authentication, and data authentication. Node authentication allows the receiver to verify if the message is sent by the claimed sensor node or not. Therefore, by applying authentication in the WSNs, an adversary will not be able to participate and inject data into the network unless it has
valid authentication keys. Alternatively, data authentication allows the receiver node to verify that the data itself was really sent by the claimed sensor node that is sending the data.

2) Access Control: This type of service prevents an unauthorized use of any of the sensor nodes.

3) Data Confidentiality: Confidentiality service ensures that data content is not revealed to an unauthorized attacker who is able to eavesdrop any of the transmitted data.

4) Data Integrity: Data confidentiality alone is not enough since an adversary can alter the data even though it knows nothing about it. The adversary is able to change the sensor reading by adding some fragments or manipulate the packet’s content without being detected before forwarding it to the next hop. Moreover, even with no adversary, data might be damaged or lost due to the unreliable wireless environment. Therefore, in WSNs, data integrity provides a strong defense against alteration of data.

5) Data Freshness: Active attackers (malicious nodes) can not only modify the data content but also delay the transmission of the captured packets and perhaps replay those packets at a later time. Data freshness ensures that the readings that are being received by the base station are fresh and untainted and no old readings have been replayed.

6) Non-Repudiation: ensures that a transferred packet has been sent and received by the node claiming to have sent and received the packet. Once the sensor node sends its reading to the base station, it should not be able to deny sending that reading.

7) Data Availability: Availability service ensures that the network is alive and that data are accessible anytime. In order for any secure mechanism to provide the availability service in the WSN, it should rely on self-healing and energy-reduction techniques. If the sensor network is self-healing, it has the ability to diagnose and react to the attacker’s activities and then start
corrective actions based on defined policies to recover the network or a node. Moreover, if the sensor network provides a mechanism for maximizing the network life time by reducing energy consumption on the sensor nodes, the network service will be available for a longer time.

One way of providing some of the above services is to use cryptography and authentication. However, as mentioned in the previous section, WSNs are known to be resource-constrained (e.g., small memory size, weak processors, limited energy, and small packet size), that means they require extra attention when applying cryptography or authentication techniques. Researchers began to design lightweight mechanisms that are suited for such networks. For instance, a package of security protocols called “SPINS” was delivered in [17]. The package consists of a lightweight cryptographic technique called “SNEP” (Secure Network Encryption Protocol) which provides the network with important baseline security primitives like data confidentiality, two-party data authentication, and data freshness, as well as another lightweight authentication mechanism called “μTESLA” (i.e., the micro edition of the Timed, Efficient, Streaming, Loss-tolerant Authentication Protocol) which provides a streaming broadcast authentication for severely resource-constrained environments.

Follows are some of the several attacks [4] targeting WSNs:

1) **DoS (Denial of Service) Attack**: A standard attack on the WSN that transmits radio signals which interfere with the radio frequencies used by the WSN, this is called “jamming”. An example of a DoS attack is when the base station is no longer able to answer the various queries.

2) **Sybil Attack** [38]: An attack where the adversary is able to present more than one node identity within the network. One example of such attack is when the adversary creates multiple identities of the sensor node to generate multiple readings which result in falsification of the resulted query.
3) **Selective Forwarding Attack:** WSNs assume that each node will accurately forward the received messages. Nevertheless, if we take security into account, a compromised node may refuse to do so. It is up to the adversary that is controlling the compromised node to either forward the received readings or not. In case of not forwarding the sensor readings, the query provided by the base station may be erroneous.

4) **Replay Attack:** In the case of a replay attack, an attacker records some traffic patterns from the network without even understanding their content and replays them later on to mislead the base station and its query answer.

5) **Stealthy Attack:** The adversary objective in this attack is to inject false data into the network without revealing its existence. The injected false data value leads to an erroneous query result at the base station.

The above mentioned attacks can be blocked using light cryptography techniques. However, what if one sensor node was physically compromised by an adversary? If this happens, all the secret keys and authentication data on that node will be easily extracted by the attacker who can launch new attacks even when those mentioned lightweight mechanisms are applied. Consequently, SPINS and other lightweight cryptographic-based security mechanisms such as TinySec [50], INSENS [51], TinyPK [52], SERP [53] and SEF [54] become ineffective in the presence of a node compromise and there is an immediate need for different security mechanisms that fight against node compromises and insider attacks.

### 1.5 Trusted Querying

The previous section focused on the significance of having a novel security mechanism other than cryptography. A careful study of trust systems introduced in the field of e-commerce leads us to think of such systems as a solution to the node compromise problem in sensor networks. In computer networks the trust is commonly referred to as belief [45] and we can
measure the level of trust as the uncertainty in belief. In Chapter III, we explain the concept of trust and provide the essential techniques for establishing trust in sensor networks.

2. Network Security Visualization

Whenever a network analyzer or administrator uses one of the existing network sniffing software tools such as Wireshark [85] to analyze the network traffic, obviously a huge amount of packets is being captured at a time and being recorded as raw texts. Exploring the traffic files would thus require a tremendous effort. Visualization can be thought of as an efficient technique that helps the network administrators observe the traffic in easier ways. What makes the story more interesting is when patterns are being captured to detect vulnerabilities in the network and further build a defense against possible attacks. Security visualization techniques have been developed over decades and are a product of much research from industry, academia and individual hacking [58]. Those techniques can be powerful when employed in the field of network security where a careful crafting of graphical windows into data can exploit the visual recognition of human eyes and leads to an early detection of malicious acts.

Table 2. Open Source Network Visualization Tools

<table>
<thead>
<tr>
<th>Tool Name</th>
<th>Development Language</th>
<th>Supported OS or Platforms</th>
<th>Data Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>rumint [69]</td>
<td>Visual Basic</td>
<td>Windows</td>
<td>Real Traffic / pcap</td>
</tr>
<tr>
<td>INAV [84]</td>
<td>Server (C++)/Client (Java)</td>
<td>Linux/Windows/Linux/Mac OS</td>
<td>Real Traffic Capture</td>
</tr>
<tr>
<td>VisFlowConnect</td>
<td>Java</td>
<td>Any</td>
<td>NetFlow (Argus, Cisco)</td>
</tr>
<tr>
<td>NVisionIP</td>
<td>Java</td>
<td>Any</td>
<td>NetFlow (Argus, Cisco)</td>
</tr>
<tr>
<td>tnv [83]</td>
<td>Java</td>
<td>Any</td>
<td>Real Traffic / pcap</td>
</tr>
<tr>
<td>InetViz [82]</td>
<td>Qt</td>
<td>Windows/Linux</td>
<td>Pcap</td>
</tr>
</tbody>
</table>
Table 2 lists some of the open-source security visualization tools developed recently. All tools in the table can run over Microsoft Windows platforms as well as several flavors of Linux except rumint [69]. Rumint, however, can be ported to Linux systems using Wine [86].

3. Thesis Contribution and Outline

Figure 2 illustrates an example setup of a web-based monitoring system for spatially temporally correlated wireless sensor networks. The system provides the end user with an online (web) querying service which retrieves the average temperature measured in the area. The main contributions of this thesis are:

- Providing correlated sensor networks with a trusted querying approach which is able to filter out untrustworthy nodes (either compromised or misbehaving nodes) and report the most-trusted query response.
- Detecting security vulnerabilities inside the network through visualizing the network traffic data.

Figure 2. A Web-based Sensor Networks Monitoring System.
The rest of this thesis is organized as follows: Chapter II discusses in-network data aggregation techniques and several schemes that build security over data aggregation. Chapter III introduces reputation-based and trust-based systems. Chapter IV details our proposed trusted querying approach for correlated WSNs. Chapter V presents our developed network security visualization tool “SecVizer”. We conclude in Chapter VI and provide some future work.
II. IN-NETWORK DATA AGGREGATION

One of the important functionalities of a sensor network is its capability of answering queries over the sensed data. Sensor-based systems are usually designed along with methods to extract useful information from the data collected by the sensors. Consequently, wireless sensor networks designers and developers initiated several data management solutions that use tiny sensor database systems to allow users to perform queries over the sensor network. Examples of such solutions are the Berkeley query processing system “TinyDB” [18] and Cougar [19] which was developed by the Cornell Database Group.

1. Overview

Perhaps the most efficient query processing technique for WSNs that maximizes the network lifetime is in-network aggregation. In-network data aggregation is the simplest form of in-network processing where the sensor nodes in the network are not just passing packets, instead, they contribute in the decision making process. The information processing is taking place in the network itself. The information is the readings of the sensor data being collected by each sensor. The aggregation of those readings forms the decision making that some sensors have to perform. By aggregation we mean the sum, average, minimum, maximum, nodes count or any other aggregation function that can be applied over the collected sensor readings. In case that the base station is interested in a specific query (say the sum of all sensor readings), it would be unnecessary to return all readings collected from each sensor node, instead, the readings are processed and aggregated by some intermediate nodes (often called aggregators) within the network and only the processed and aggregated data is returned. For the purpose of network lifetime maximization, in-network data aggregation reduces the number of packets being transmitted within the network. Figure 3 illustrates the procedure, in (a) no aggregation is applied at the intermediate nodes, as a result each one of those nodes has to forward the readings that it
receives from the neighboring nodes to the next hop ending with the gateway that collects all 
those readings and performs the aggregation function; the number of the data packets being 
transmitted through the network is 29 packets. However, in (b) the intermediate nodes perform 
the desired aggregation function to calculate the result queried by the gateway and hence only 
the resulted packet will be transmitted through the wireless link to the next hop (no need to 
forward all readings received by the neighboring nodes). The number of data packets being 
transmitted in this case is 16 packets.

We can clearly conclude that since the sensor power usage is largely determined by the 
transmission cost, the transmission of less data (transmitting the result of the aggregation instead 
of forwarding all the packets) reduces the energy consumption at the sensor nodes. It also 
reduces the congestion in the network as well as the collision of packets or the packet loss and 
thus avoiding retransmission which consumes extra energy.

![Figure 3. Efficiency of In-Network Data Aggregation](image)

Let’s check if this in-network data aggregation mechanism satisfies the design 
characteristics mentioned in the previous chapter. Generally, intermediate nodes do not store any 
of the readings received neither the aggregation result. This satisfies the low storage capability
requirement. In most of the cases, the intermediate nodes also do not perform complex computations on the collected sensor readings; all they do is summing, averaging, minimizing or maximizing those readings. These operations are considered lightweight operations on the sensor and do not require high computational power. Finally, the reduction in the number of packets being transmitted will satisfy the communication overhead requirement.

2. Aggregation Schemes

Many data aggregation techniques have been proposed for WSNs. A very well-organized and almost complete survey of the several in-network data aggregation schemes has been formed in [10]. It studied tree-based schemes, cluster-based schemes, multi-path schemes as well hybrid schemes that make benefit of both the tree-based and the multi-path approaches for data aggregation. In this section, we will have a quick look on the most popular mechanisms; the first four were discussed in [10], each has a different way of achieving in-network aggregation. They generally fall under one of the following categories: tree-based, cluster-based, multi-path and a hybrid scheme that combines both tree-based and multi-path approaches together.

2.1 TAG

TAG (Tiny AGgregation) [11] is a tree-based aggregation scheme. Tree-based schemes provide the simplest way of achieving data aggregation. The procedure looks the same as in Figure 3 (b). The sink broadcasts a message asking nodes to organize into a routing tree and then sends its queries. After the construction of the tree, the queries are sent along the structure to all nodes in the network. During the data collection phase, each intermediate node has to wait for data from all of its children before it can send its aggregate up the tree and data aggregation is performed by all intermediate nodes. In practice, a node goes back to sleep soon after it has finished sending its readings to its parent thus saving some energy in addition to the reduction of energy needed for retransmitting packets when dropped in case of no aggregation applied.
One of the drawbacks of such scheme is its inefficiency in case of dynamic topologies or link/device failures: trees are particularly sensitive to failures at intermediate nodes as the related sub-tree may become disconnected. In addition, as the topology changes, TAG has to re-organize the tree structure and this means high costs in terms of energy consumption and overhead.

2.2 LEACH

LEACH (Low-Energy Adaptive Clustering Hierarchy) [12] is a cluster-based aggregation scheme that is similar to tree-based schemes because the network is also hierarchically organized. However, nodes are subdivided into clusters. Also, special nodes, referred to as cluster-heads, are elected in order to aggregate data locally and transmit the result of such an aggregation to the sink. Figure 4 shows four clusters with four cluster heads being elected by each cluster’s sensor nodes. The advantages and disadvantages of cluster-based schemes are very similar to those of tree-based approaches.

This scheme is adaptive which uses randomization to evenly distribute the energy expenditure among the sensors. Clustered structures are exploited to perform data aggregation where cluster-heads act as aggregation points. It employs the TDMA protocol in the data collection phase to ensure that there are no collisions within the clusters, saving both energy and time. It also implements a doze mode to further save energy. When doze mode is used, the nodes’ radios may be switched off until their scheduled TDMA transmission slot. Note that cluster-heads cannot switch their radio off as they have to receive packets from potentially all nodes in the cluster. Mobility results in additional problems where a node close to a cluster-head at a given instant in time may move away from the cluster-head. As a consequence, the node needs to increase its power, thereby spending much more energy to transmit to the cluster-head than expected.
2.3 Synopsis Diffusion

Hierarchical schemes are inefficient when a node failure is present. Imagine the node that fails is the one that is a direct child to the sink, the whole aggregate result of the sub-tree (with the failed node being its root) is lost. To solve this issue, Synopsis Diffusion [13] has been proposed. Synopsis diffusion achieves significantly more accurate and reliable query answers by combining energy-efficient multi-path routing schemes with techniques that avoid double-counting. Figure 5 illustrates a ring overlay. Nodes are arranged into rings (R₀, R₁ and R₂) and receive readings from different paths. Even though there are link and node failures, nodes A and B have at least one failure-free propagation path to the base station (the querying node). Thus, their sensed values are accounted for in the final answer. In addition to the high fault-tolerance, this scheme also provides a solution to the problem of duplicate sensitivity which is a property of some aggregation functions such as SUM by using order- and duplicate-insensitive (ODI) synopses that compactly summarize intermediate results during in-network aggregation. In the
absence of ODI, an intermediate node will receive readings from multiple children and each of
those received sensor readings will be accounted for as a new reading.

2.4 Tributaries and Deltas

A hybrid scheme in [14] combines both the tree-based approach along with the multi-path
approach. By doing this, it overcomes the problems of both structures. In case of low packet loss,
the nodes perform as if they are in a tree-based structure whereas in case of high packet drop
ratio, the nodes will switch to the multi-path structure.

2.5 CountTorrent

Synopsis diffusion performs well in a mobile environment. However the accuracy of the
aggregate result is not high. Another scheme that performs well in the presence of mobility is
called CountTorrent [15]. This scheme remains efficient and accurate even as nodes move, join
or leave the network. In case of stationary networks, it has a 100% accuracy in the aggregate
result even in the presence of lossy links while it provides a close (within 10-20%) estimate of
the accurate aggregate query value to all nodes in the network at all time.
2.6 Approximate Aggregation Techniques

The drawback of the synopsis diffusion scheme is its inefficiency in the presence of duplicate sensitive aggregates. [16] solves the problem of duplicate sensitivity using approximate in-network aggregation using small sketches. This scheme exploits the sketch theory to compute approximates for the duplicate sensitive aggregation functions such as network count (i.e. number of nodes in the sensor network), summation, average which can be computed directly from the count and the sum sketches. The scheme also provides a method for combining both duplicate insensitive sketches together with multi-path routing techniques to produce more accurate approximations.

Table 3 provides a comparison of the discussed schemes. One thing to notice is the extra energy saving mechanisms that both TAG and LEACH use, which the other schemes lack. You can also notice that the accuracy of CountTorrent in presence of mobility is the highest compared with others. Also, CountTorrent has the lowest overhead to maintain the aggregation structure.

<table>
<thead>
<tr>
<th>Aggregation Method</th>
<th>TAG</th>
<th>LEACH</th>
<th>Synopsis Diffusion</th>
<th>Tributaries and Deltas</th>
<th>CountTorrent</th>
<th>Approximate Aggregation Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resilience to link failures</td>
<td>Medium</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Accuracy in case of node mobility</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
<td>Very High</td>
<td>High</td>
</tr>
<tr>
<td>Overhead to setup/maintain the aggregation structure</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Scalability</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Energy saving Methods</td>
<td>Sleeping periods</td>
<td>Local route repaires</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
3. Secure Data Aggregation

As being discussed earlier, designing a data aggregation mechanism for wireless sensor networks is very challenging. What makes it more challenging is when the sensor nodes are deployed in a hostile environment where they are very likely to be vulnerable to node compromise by an insider attacker. An adversary might appropriate a regular sensor node and inject false data into the WSN. The scenario is worsened when that sensor node is the node that performs the aggregation. The adversary can alter the entire aggregate result and pervade the network with falsified results. Physical tampering, thus, created a new challenge in sensor networks and began to attract more and more attention. Manufacturers who were aware of such issue tried to provide the wireless sensors with tamper-resistant hardware. However, since sensor nodes are envisioned to be tiny little devices with low-cost, this solution becomes infeasible.

Data aggregation itself requires specialized security services such as data integrity, data confidentiality, node authentication and data freshness. One way to embody the latter services into data aggregation is to use cryptography. However, as mentioned in the previous chapter, when designing a cryptographic technique for data aggregation we should consider the impact of the added security features on the low energy consumption and all other design limitations. Schemes designers should also take into consideration the adversarial model [22] they are dealing with which includes the type of the adversary (passive or active), the type of network access (total access or partial access) as well as the type of access of the secret key (total vs. partial). In fact, a conceptual scheme evaluation framework has been proposed in [26] which helps the new security schemes designers strengthen their proposed scheme against the various adversarial models. [26] also surveys the existing state-of-the-art secure data aggregation schemes. These schemes were classified into two groups according to the number of aggregator nodes and whether the integrity of the aggregated result is considered or not. Some of the
schemes discussed in the survey were SIA [27] and SDA [28]. Those schemes provide cryptographic solutions over the tree-based aggregation schemes (TAG, LEACH). Alternatively, to secure the process of Synopsis Diffusion, [29] has proposed an attack-resilient aggregation scheme over a multi-path environment which also uses MACs (Message authentication codes) to verify the validity of the synopses contribution to the aggregate function at the sink.
III. TRUST MANAGEMENT IN SENSOR NETWORKS

The discussion in the previous chapters (security in WSNs in Chapter I, secure data aggregation in Chapter II) concentrated on the significance of discovering solutions to the problem of node compromise. The impact of malicious attacks on wireless sensor networks has been extensively studied in [4] [38] [40] [41]. As mentioned before, several proposals (such as SPINS), all based on cryptography, have been initiated to ensure secure communication on these resource constrained sensor nodes. The establishment and management of the cryptographic keys [17] [53] [55] [56] [32] form the backbone of these schemes; however, the scale and ad-hoc deployment of nodes coupled with the ability of adversaries to easily recover the cryptographic materials make countering node compromise and ensuring trustworthiness in WSNs a challenging problem to solve.

Based on this, WSN security researchers began to explore solutions other than the pure cryptographic solution. These new solutions borrow tools from different domains such as economics, statistics, machine learning, and data analysis and combine them with cryptography for the development of trustworthy sensor networks. In the following section we define two very useful concepts that are used in facilitating decision making in diverse fields and mainly in e-commerce (reputation and trust). Section 2 provides the schemes’ designers with essential trust establishment techniques. We discuss some of the most popular attacks on the reputation and trust-based frameworks in sensor networks in Section 3.

1. Reputation and Trust Definition

In social science, reputation is defined as the perception that a person/party has of another’s intention. In computer networks, reputation is the opinion of one entity about another. In an absolute context, it is the trustworthiness of an entity [42]. On the other hand, trust in social science is identified by several representative trust constructs [44]. In computer networks, there
is not yet a clear consensus on the definition of trust. [45] identified two main constructs of the trust concept that are built upon a belief formulation process; trusting belief and system trust. [45] refers to the three models (belief formulation process, trusting belief and system trust) as trust management.

![Figure 6. Trust Constructs in Computer Networks](image)

Figure 6 shows the representative constructs in computer networks suggested by [45]. The outcome of trust management is provided to decision making functions, which will make decisions based on trust evaluation as well as other application-related conditions. Furthermore, system trust can be interpreted as a special type of belief, where an entity believes that the network will operate as it is designed. Thus, belief is the most appropriate interpretation of trust in computer networks. One entity believes that the other entity will act in a certain way, or believes that the network will operate in a certain way.

2. Trust Establishment

In computer networks, there are two common ways of establishing trust [46] either directly or indirectly through a recommender. Direct trust is established upon observations on whether the previous interactions between two nodes A and B are successful and is denoted by $T_{AB}^d$. A special case of direct trust is the recommendation trust where node A can judge whether a
recommendation about B is correct or not. Recommendation trust is denoted by $T'_{AB}$. On the other hand, indirect trust establishment is obtained by transiting trust through third parties, a phenomenon called trust propagation. For instance, if node A and B have established a recommendation trust relationship and node B and C have established a direct trust relationship, then node A can trust node C to a certain degree if node B tells A its trust opinion (i.e. recommendation) about node C. A trust relationship means that one party trusts the other party to perform a specific action.

![Figure 7. Trust Propagation for Indirect Trust Establishment](image)

There are two key factors to determine the indirect trust establishment in computer networks. First, a recommendation mechanism determines the recommenders and when to collect recommendations. Second, determine how to calculate indirect trust values based on recommendations. Trust models are used for the latter purpose and usually include the concatenation model and the multi-path model. Figure 7 illustrates the concept of trust propagation in establishing indirect trust in a network of four nodes A, B₁, B₂ and C. Node B₁ and node B₂ observe the behavior of node C and both establish direct trust in C with trust values $T^d_{B_1C}$ and $T^d_{B_2C}$ respectively. Node A has recommendation trust in both B₁ and B₂ with trust values $T'_{AB_1}$ and $T'_{AB_2}$.
Node B1 and node B2 provide recommendation about C by telling A the values of $T_{AB_1}^r$ and $T_{AB_2}^r$. The concatenation model is a function that calculates the indirect trust values between A and C from $T_{B_1C}^d$ and $T_{AB_1}^r$ through the recommender node B1 and from $T_{B_2C}^d$ and $T_{AB_2}^r$ through the recommender node B2. The concatenation function is denoted by $f_{ctp}(.)$, whereas multi-path model is a function that combines trust established through multiple paths and is denoted by $f_{mtp}(.)$. The final indirect trust value is denoted by $T_{AC}^{ind}$ and is calculated as follows,

$$T_{AC}^{ind} = f_{mtp}(f_{ctp}(T_{B_1C}^d, T_{AB_1}^r), f_{ctp}(T_{B_2C}^d, T_{AB_2}^r))$$

Trust propagation is governed by three axioms; 1) Concatenation propagation of trust does not increase trust, 2) Multi-path propagation of trust does not decrease the trust value and 3) The recommendations from independent sources can reduce uncertainty more effectively than the recommendations from correlated sources (i.e. trust based on multiple recommendations from a single source should not be higher that that from independent sources).

In Chapter IV, we derive trust from parallel transitive paths using subjective logic. The idea is similar to establishing indirect trust relationships by applying the concatenation and multi-path models.

3. Attacks on Reputation and Trust-Based Schemes

Although trust-based schemes (e.g. RFSN [41], [49]) play an effective role in detecting malicious nodes in the sensor network, they themselves attract attackers and are vulnerable to attacks. In this section we discuss four common attacks [47] that target trust-based frameworks and provide a defense against them whenever possible.
3.1 Bad Mouthing Attack

The bad mouthing attack is the most straightforward attack and has been discussed in many existing trust management or reputation systems. It occurs when malicious parties provide dishonest recommendations [48] to frame up good parties and/or boost trust values of malicious peers.

The defense against this attack has three perspectives [45]. First, only the nodes who provided good recommendations previously can earn high recommendation trust. Second, recommendation trust plays an important role in the trust propagation process. The necessary conditions of trust propagation state that only the recommendations from the nodes with positive trust values can propagate. In addition, the trust propagation axioms limit the recommendation power of the entities with low recommendation trust. Third, the recommendation trust is treated as an additional dimension in the malicious node detection process. As a result, if a node has low recommendation trust, its recommendations will have minor influence on good nodes’ decision-making, and it can be detected as malicious and expelled from the network.

3.2 On-off Attack

In this attack the malicious nodes behave well and badly alternatively, hoping that they can remain undetected while causing damage. Trust is dynamic in nature which means that a good node may be compromised and turned into a malicious one, while an incompetent node may become competent due to environmental changes. This attack exploits the dynamic properties of trust through time-domain inconsistent behaviors. To track this dynamics, the observation made a long time ago should not carry the same weight as that made recently.

The defense against the on-off attack is through introducing an adaptive forgetting factor. The idea is inspired by the social phenomenon that a human remembers bad behaviors for a longer time than for good behaviors. By using the adaptive forgetting factor, the trust value can
keep up with the node’s current status after the node turns bad while a node can recover its trust
value after bad behaviors, a recovery that requires many good actions.

3.3 Conflicting Behavior Attack

In the on-off attack, the attacker behaves inconsistently in the time domain. In the
conflicting behavior attack, on the other hand, the attacker behaves inconsistently in the user
domain. In particular, malicious nodes can impair good nodes’ recommendation trust by
performing differently to different peers. For example, the attackers can always behave well to
one group of nodes and behave badly to the other group and therefore, these two groups develop
conflicting opinions about the malicious nodes. Nodes in the first group obtain recommendations
from the other group, but those recommendations will not agree with the first group’s own
observations. As a consequence, the users in one group will assign low recommendation trust to
the users in the other group.

3.4 Sybil Attack and Newcomer Attack

A trust management system may suffer from the sybil attack [38] when a malicious node
can create several faked IDs. The faked IDs can share or even take the blame, which should be
given to the malicious node. On the other hand, a trust management system may suffer from the
newcomer attack [39] when a malicious node can easily register as a new user. Malicious nodes
can easily remove their bad history by registering as a new user. The new comer attack can
significantly reduce the effectiveness of trust management.

The defense against the sybil attack and newcomer attack does not rely on the design of
trust management, but the authentication schemes. Authentication is the first line of defense that
makes registering a new ID or a faked ID difficult.
IV. TRUSTED QUERY IN SENSOR NETWORKS

Chapter III gives an overview on the use of reputation and trust in designing secure mechanisms for sensor networks. In this chapter, we design and demonstrate the feasibility of an innovative reputation-based framework rooted in rigorous statistical theory and belief theory to characterize the trustworthiness of individual nodes in a wireless sensor network (WSN). The resulting mechanism allows the detection of compromised nodes as well as misbehaving nodes. Moreover, trusted querying is enabled by filtering out “untrustworthy sensor nodes and data” and returning the most-trusted aggregate response. We showcase the effectiveness of the proposed framework through a simulation based study.

1. Introduction

As discussed in the previous chapters, security breach can happen in a WSN not only while relaying information to the end-user but also while generating information where the problem is to deal with manipulation of the environment or the sensing channel for cheating and attacks on the integrity of sensing. The traditional approach of providing network security has been to borrow tools from cryptography and authentication. Cryptography presents mechanisms for providing data confidentiality, data integrity, node authentication, secure routing and access control. However, cryptography alone is not sufficient. Attaching message authentication codes (MACs) can verify the consistency of data but cannot verify its validity as the source generating the data itself can be malicious.

On the other hand, sensor nodes are very likely to be deployed in hostile environments. As long as sensor nodes are envisioned to be low-cost, it would be infeasible for manufacturers to make them tamper-resistant. Therefore, they can be compromised, and an adversary can then launch attacks upon recovering the secret key. A few recent research efforts have proposed mechanisms to provide authentication for wireless sensor networks to prevent false data injection
by an outsider attacker [28], [30], [31]. Their basic approaches [3] for security are to use MACs and probabilistic key pre-distribution schemes such as those proposed in [32], [33]. These approaches prevent naive impersonation of a sensor node; however, they cannot prevent the injection of forged or false data from malicious or compromised insider nodes, which have already been authenticated as legitimate ones in the networks. Once authenticated as a legitimate node, broadcasting data from that node will be accepted as trusted data in the networks. Besides malicious security breaches, bogus data can also be generated by nodes unintentionally due to the failure of some system components such as radios, sensors etc.

Conventional view of security based on cryptography [3] alone is thus no longer sufficient for the unique characteristics and novel misbehaviors encountered in wireless sensor networks. Fundamental to this is the observation that cryptography cannot prevent malicious or non-malicious injection of data from internal adversaries or misbehaving nodes. Therefore, the ability of a wireless sensor network to perform its task depends not only on its ability to securely communicate among the nodes, but also on its ability to securely sense the physical environment and collectively process the sensed data. This decentralized in-network decision-making, which relies on the inherent trust among the nodes [34] [35] [36] [37], can be abused by adversaries to carry out security attacks through compromised nodes. Dealing with insider attacks (such as those caused by node compromise) and node misbehavior has been a great challenge in resource constrained wireless sensor networks. Ultimately, from the perspective of a sensor network end-user, a secure WSN should provide trustworthy services, such as supporting trusted querying.

To this end, we believe that, generally, tools from different domains such as economics, statistics, machine learning, and data analysis will have to be combined with cryptography for the development of trustworthy sensor networks. Following this approach, we propose a
reputation-based spatial temporal correlated sensing framework (Figure 8) rooted in statistical theory, reputation, trust, as well as belief modeling for building wireless sensor networks. In this framework, nodes maintain reputation of other local nodes, and use reputation to evaluate their trustworthiness. We demonstrate the feasibility of this mechanism to characterize the trustworthiness of individual nodes in a wireless sensor network. The resulting mechanism allows the detection of compromised nodes and misbehaving nodes. Moreover, trusted querying is enabled by filtering out “untrustworthy sensor nodes and data” and returning the most-trusted aggregate response. Finally, we showcase the proposed mechanism through a simulation based study.

The rest of this chapter is organized as follows. Section 2 presents the reputation-based spatial temporal correlated sensing framework. Section 3 describes sensor node reputation characterization and update scheme. Section 4 details sensor node classification and compromised node detection. Aggregation result uncertainty quantification is given in Section 5. The results of simulation based evaluation are reported in Section 6. We summarize the chapter in Section 7.

2. Reputation-based Spatial Temporal Correlated Sensing Framework

We consider a sensor network composed of a large number of densely deployed sensors that are organized into clusters using clustering schemes such as LEACH [12]. Sensor nodes can also be clustered based on geo-proximity. Figure 8 schematically illustrates the architecture of the proposed reputation-based spatial temporal correlated sensing framework.

Within each cluster, nodes are divided into a number of separate aggregation sets. Each aggregation set has an elected aggregator. The number of aggregation sets depends on the cluster’s density and desirable data accuracy. In a cluster, all sensor nodes including the cluster head and aggregators are physically proximate. The framework takes advantage of the fact that
sensory data are spatially and temporally correlated for sensor node reputation characterization and compromised node detection. A cluster head acts as a gateway of the cluster to the base station, and responds to end-user queries (periodical or on-demand) by sending the queries to individual aggregators. Aggregators, in turn, sample individual sensor nodes for data and return aggregate responses to the cluster head which then combines the responses from aggregators to form an answer to the end-user query and forwards it to the base station. For ease of exposition, we assume that each sensor has bidirectional communication capability and can directly communicate with its cluster head. Each time an aggregator integrates, all the reported data from sensor nodes within its aggregation set constitutes a sampling round.

Figure 8. A Schematic Illustration of a Reputation-based Spatial Temporal Correlated Sensing Framework.
The threat model that we consider assumes that an adversary can compromise any sensor node including the cluster head and aggregators. These compromised nodes have the same computation and communication capability as those of the normal nodes. An adversary can manipulate the compromised nodes and alter/forged sensory data to disrupt normal network operations while circumventing cryptography and authentication approaches aimed at guaranteeing data integrity or secrecy. Our threat model is also general enough to consider that a compromised node can inject dramatically different data from the true sensor readings, or inject covert data that do not apparently deviate from the true sensor readings but can intentionally influence the outcomes of the sensor network and responses to the end-user queries in the long run. The latter case may easily evade the detection of sophisticated security measures and therefore is more dangerous and difficult to deal with.

As a first line of defense, encryption and authentication schemes are employed to enable secure information exchange within a cluster, such as the election of the cluster head, aggregators, and broadcast of the election results to all sensor nodes within the cluster. To prevent impersonation, a broadcast authentication technique, such as µTESLA [17], can be employed. The sensory data from each sensor node is protected by a MAC with the pairwise key shared between the node and its aggregator. Our innovative contribution is to build a level of trust into the system based on rigorous statistical theory and belief theory that utilize behavior relationships between neighboring nodes, and as a result, enable the capability of detecting compromised nodes and filtering out these nodes and untrustworthy data when responding to end-user queries.

Within an aggregation set, the aggregator maintains and updates reputation of each sensor node that represents this node’s trustworthiness. Reputation is defined as the perception that a
person has of another’s intention. Trust is viewed as belief that one entity believes that the other will act in a certain way, i.e., it describes the level of uncertainty in trust relationship. The reputation metric is constructed based on the statistical properties or observation consistency of sensory data. When a sensor node produces sensory data with statistical properties that are deviated from the norm, its reputation is considered tarnished. Accordingly, this node becomes less trustworthy. After collecting sensor data from each node, an aggregator first classifies these nodes into different groups based on their reputation. The aggregate result of the aggregation set is calculated based on the sensor data from the group of nodes with the highest reputation. Each sensor node’s reputation is then updated by comparing its sensory data with the aggregation result. Based on Josang’s belief model [20], by examining the aggregation result and sensor nodes’ reputation, the aggregator further formulates an “opinion” (details of which are given below) of the aggregation result. The opinion measures the uncertainty inherent in the aggregation result, and it expresses the aggregator’s degree of belief regarding the truthfulness of this result. The aggregator reports the aggregation result and associated opinion to the cluster head. The cluster head in turn integrates the aggregation results from multiple aggregators and associated opinions to derive at a final query response which is sent to the base station. At the same time, all sensor nodes can overhear the reports sent by the aggregators and the cluster head so that they can evaluate and update the reputation of the aggregators and the cluster head based on their own judgment.

Our framework enables each node to build up reputation based on its behavior over time. Compromised nodes can be detected by checking their reputation. A new aggregator or a new cluster head can be re-elected using nodes’ reputation information if needed (e.g., when they become compromised or misbehave due to faults, or for the purpose of balancing each node’s
resource use such as power, the role of aggregators and cluster head must be rotated as in LEACH [12]). The opinions of aggregation results can be propagated throughout the network from aggregators, cluster head, and/or an ad-hoc network, and eventually to base stations. This aggregation result and opinion propagation process is governed by a set of subjective logic rules [21] [22].

3. Sensor Node Reputation Characterization and Update

Central to our framework is the characterization of reputation of individual sensor nodes and the derivation of a meaningful and powerful trust metric from reputation. Moreover, reputation characterization should be grounded on a solid statistical or information theoretic basis. Through close local interaction among sensor nodes, aggregators, and cluster heads, reputation of nodes are built over time and cross monitored to provide checks and balances.

Most sensor network applications are based on local interactions between nodes that typically lie in the neighborhood of each other. To the best of our knowledge, there exists no sensor network application whereby a node will require prior reputation knowledge about a node many hops distant from it. We note that even if in the future some applications require instant reputation information of a distant node, it can be established dynamically at runtime using the chain of trust relationships between neighboring nodes. In our framework, nodes maintain reputation information only about its neighboring nodes, i.e. nodes that lie in its broadcast domain. This property of “locality” holds the key for scalability of sensor networks. This same property substantiates our claim of developing a reputation-based framework for trustworthy sensor networks. Not only the nodes need to maintain reputation and trust metrics for only a few nodes in the network but they can also easily establish this metric quickly through local interaction.
Specifically, reputation is defined as the perception that a person/party has of another’s intention. Trust is the extent to which one person/party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible. It is used by the person/party to make a choice, when an action must be taken before the actions of others are known [23]. When facing uncertainty, individuals tend to trust those which have a reputation for being trustworthy [23]. A framework based upon reputation and trust will help the nodes to distinguish good nodes from bad. Therefore, it is critical to reliably characterize a sensor node’s reputation. Note that reputation is not a physical quantity but it is a belief; it can only be used to statistically predict the future behavior of other nodes and cannot define deterministically the actual action performed by them. We develop two types of reputation characterization and update schemes.

3.1 Relative entropy based scheme

The idea of this information-theoretic approach is to extract the underlying statistical characteristics from sampled data (i.e., sensor readings) over time and exploit such information to evaluate each sensor node’s reputation. In probability theory and information theory, the relative entropy is a measure of the difference between two probability distributions: from a “true” probability distribution $P$ to an arbitrary probability distribution $Q$. Typically $P$ represents data, observations, or a precisely calculated probability distribution. The measure $Q$ typically represents a theory (ideal), a model, a description or an approximation of $P$. For probability distributions $P$ and $Q$ of a continuous random variable, the relative entropy of $Q$ from $P$ is defined as

$$D(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} \, dx.$$
The relative entropy can be considered as a “distance” between the probability distribution of sampled sensory data over time and the “ideal sensory data.” Intuitively, the shorter the “distance” the closer the sensory data is to the ideal data, which means that the node that is generating the data is less likely to have been compromised or misbehaving and is therefore more reputable. The reputation of the node can then be defined as being inversely proportional to a function of $D(P \parallel Q)$, e.g.,

$$\frac{1}{1 + f(D(P \parallel Q))} \in [0,1],$$

where $f(\bullet)$ is a smoothing function. Note that when $P$ is the same as $Q$, $D(P \parallel Q) = 0$ and the reputation of the sensor node is 1 (i.e., perfect reputation). This type of scheme depends on knowing the ideal probability distribution of sensory data and query type. Surprisingly, for many types of query, this approach is indeed feasible, such as determining the means of sensory data as demonstrated in our simulation study.

### 3.2 Consistency based scheme

This statistical approach is based on Bayesian formulation. Each node maintains reputation of its neighbors. A node updates its neighbor’s reputation based on whether or not the latter’s data observed is consistent with its own sensory reading. Several distributions such as beta, Gaussian, Poisson, binomial, can be used to represent the reputation of a node. The beta distribution has been determined to be flexible and simple as well as being strongly rooted in the theory of statistics. In particular, a beta reputation system has been proposed and analyzed in [20].

The beta distribution of $x$ is indexed by two parameters $(\alpha, \beta)$. It can be expressed using the gamma function as:

$$P(x) = Beta(\alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1-x)^{\beta-1}, \forall 0 \leq x \leq 1, \alpha \geq 0, \beta \geq 0.$$
(a) Beta(1,1) = Uniform Distribution

(b) Beta(8,2)

Figure 9. Examples of a Beta Distribution [20]

Due to the generally assumed broadcast nature of wireless sensor nodes, a node checks the consistency of data observed by a neighboring node when it reports the sensory data to the
aggregator. A simple comparison will result in a binary outcome (i.e., consistent being 1 while inconsistent being 0). The definition of being consistent or inconsistent is application dependent. We will constrain ourselves to binary outcomes only although a more generalized non-binary outcome can be considered. Reputation characterization of a node mounts to predict the future behavior of the node. Assume that node $i$ has observed node $j$ $m + n$ times; out of which $m$ times the outcome is consistent and $n$ times the outcome is not. Given this information node $i$ wants to predict the behavior of node $j$, i.e., the probability of outcome being consistent, $P(x)$ for the next observation. Without any a priori information, $x$ is uniformly distributed. Thus $P(x) = Beta(1,1)$ (Figure 9 (a)). We can model the prior outcomes using a binomial distribution and then the posteriori distribution of $x$ can be derived as: $P(x) = Beta(m + 1, n + 1)$ (e.g., $Beta(8,2)$ in Figure 9 (b)). Therefore, the beta distribution provides a simple closed form result. The beta function is the conjugate prior for the binomial likelihood distribution. This implies that if the a priori distribution is the beta distribution and the new observations follow a binomial distribution, then the posteriori distribution will also be a beta distribution. Given node $i$’s reputation $P(x)$, node $i$ again makes $r + s$ observations of node $j$ with $r$ outcomes being consistent and $s$ outcomes being inconsistent. The reputation of node $i$ can be updated as $P(x) = Beta(m + r + 1, n + s + 1)$.

4. Sensor Node Classification and Compromised Node Detection

After the reputation of nodes becomes available, the aggregator can use different ways to identify compromised nodes. A straightforward approach is to use a predefined threshold. If a node’s reputation is below this threshold, the node is considered as compromised. However, determining a proper threshold is challenging. In addition, the threshold should be adaptive in order to take into account the dynamics of the WSN. Note that a compromised node may even
launch attacks (e.g., badmouthing attacks) to ruin the reputation of a legitimate node, therefore reducing the reputation of the node. We also observe that in the long term, all the legitimate nodes have higher reputation than compromised nodes as long as compromised nodes do not dominate because the reputation of a node is built over time based on inherent statistics followed by most nodes. Therefore, nodes with different levels of reputation tend to cluster together and can thus be partitioned into separate groups.

We design a clustering algorithm to partition nodes based on node reputation into groups so that the pairwise dissimilarities between those assigned to the same cluster tend to be smaller than those in different clusters. The $K$-means algorithm is an algorithm to cluster objects based on attributes into $K$ partitions and attempts to find the centers of natural clusters in the data [24]. The objective that it tries to achieve is to minimize total intra-cluster variance, or, the squared error function

$$V = \sum_{i=1}^{K} \sum_{x_j \in S_i} (x_j - \mu_i)^2$$

where there are $K$ clusters $S_i, i = 1, 2, \cdots, K$ and $\mu_i$ is the centroid or mean point of all the points $x_j \in S_i$. We adapt the $K$-means algorithm to iteratively determine the number of natural partitions $K$. This can be accomplished by examining the within-cluster dissimilarity $V$ as a function of $K$. As $K$ increases, $V$ generally decreases and tend to decrease substantially with each successive increase in the number of specified clusters as the natural groups are successively assigned to separate clusters. When the number of clusters $> K$, one of the estimated clusters must partition at least one of the natural groups into two sub-groups. This will tend to provide a sharply smaller decrease in $V$ as $K$ is further increased, and therefore provide a stopping criterion. Once the nodes in an aggregation set are classified into different groups based on their reputation attained, the aggregator is able to detect and identify potential compromised
nodes because compromised nodes can only affect the number of partition groups. As an example for implementing trusted query processing, the aggregator can collect data by nodes from the highest reputation group and respond. By only considering the data from highest reputation group, aggregation results are immune to the influence asserted by compromised nodes with low reputation.

5. Aggregation Result Uncertainty Quantification

To enable trusted querying, we need to quantitatively gauge the level of uncertainty in a returned response. Our approach is based on belief theory. Belief theory is a framework related to probability theory, but where the probabilities over the set of possible outcomes not necessarily add up to 1, and the remaining probability is assigned to the union of possible outcomes. Belief calculus is suitable for approximate reasoning in situations of partial ignorance regarding the truth of a given proposition. Specifically, we borrow Josang’s belief model [20] to explicitly quantify the uncertainty in sensory data aggregation because data received through sensors are inherently noisy and unreliable due to the unavoidable sampling errors, false data injected by compromised nodes, misbehaving nodes, or aggregators.

Josang’s belief model proposes a belief metric called opinion to express the degree of belief in the truth of a statement. Considered as part of the subjective logic [21] [22], subjective opinions express subjective beliefs about the truth of propositions with degrees of uncertainty. An opinion is denoted as $\omega_x^A = (b,d,u,a)$ where $A$ is the subject; $x$ is the proposition (or result) to which the opinion applies; $b$ (belief) is the belief that the specified proposition is true; $d$ (disbelief) is the belief that the specified proposition is false; $u$ (uncertainty) is the amount of uncommitted belief; and $a$ is the $a$ priori probability in the absence of evidence about the subject. Furthermore, $a,b,d,u \in [0,1]$ and $b + d + u = 1$. The probability expectation value of an
opinion is defined as \( O = E(\omega) = b + au \). In the absence of any specific evidence about a given party, the base rate \( a \) determines the \textit{a priori} trust that would be put in any member of the community. An opinion where \( b = 1 \) is equivalent to binary logic \text{TRUE}, where \( d = 1 \) is equivalent to binary logic \text{FALSE}, where \( b + d = 1 \) is equivalent to a traditional probability. Therefore, \( a \) determines the degree that uncertainty \( u \) contributes to \( E(\omega) \).

![Figure 10](image.png)

Figure 10. An Example where the Opinion about a Proposition \( x \) from a Binary State Space Has the Value \( \omega_x = (0.7, 0.1, 0.2, 0.5) \) \([20]\)

The opinion space can be mapped into the interior of an equal-sided triangle, where, for an opinion, \( \omega^x = (b, d, u, a) \) the three parameters \( b, d, u, \) determine the position of the point in the triangle representing the opinion. Figure 3 illustrates an example where the opinion about a proposition \( x \) from a binary state space has the value \( \omega_x = (0.7, 0.1, 0.2, 0.5) \). The top vertex of the triangle represents uncertainty; the bottom left vertex represents disbelief, and the bottom right vertex represents belief. The parameter \( b \) is the value of a linear function on the triangle which takes value 0 on the edge which joins the uncertainty and disbelief vertices and takes value 1 at the belief vertex. In other words, \( b \) is equal to the quotient when the perpendicular distance
between the opinion point and the edge joining the uncertainty and disbelief vertices is divided by the perpendicular distance between the belief vertex and the same edge. The parameters $d$ and $u$ are determined similarly. The base of the triangle is called the probability axis. The base rate is indicated by a point on the probability axis, and the projector starting from the opinion point is parallel to the line that joins the uncertainty vertex and the base rate point on the probability axis. The point at which the projector meets the probability axis determines the expectation value of the opinion, i.e. it coincides with the point corresponding to expectation value $E(\omega)$. Using Josang’s belief model, an aggregator can formulate an opinion as well as a probability expectation value about the aggregate result.

By introducing opinion as a subjective belief to interpret the degree of trust about aggregate results and applying subjective logic [22] on the opinions to manage trust propagation from sensor nodes through the sensor network (i.e., sensor nodes, aggregator, cluster head, and other ad-hoc WSN nodes along the path to the base station), the uncertainty in the query response can be precisely quantified, which offers a handle on measuring “most trusted” query responses.

Specifically, consider two parallel transitive paths (sensor nodes, aggregator, and cluster head) as in Figure 11. Cluster $C$ receives aggregate results from aggregators $A$ and $B$ with opinions $\omega^A_x$ and $\omega^B_x$, respectively. At the same time, cluster $C$ maintains reputation and corresponding opinions about aggregators $A$, $\omega^C_A$, and $\omega^C_B$, respectively, using consistency based scheme developed in Section 3. When cluster $C$ formulates an opinion about aggregation result from two parallel transitive paths, it needs to take into account its own opinion about the aggregators.
Intuitively, if cluster C does not have a high confidence of an aggregator, then the aggregation result from this aggregator should be discounted. Therefore, using subjective logic, belief discounting can be used to compute trust transitivity along a path. For example, given $\omega^A_x, \omega^B_x, \omega^C_y$ and $\omega^C_y$, cluster C generates a discounted opinion about the aggregation results $\omega^{CA}_x = (b^C_A b^A_x, b^C_A d^A_x, d^C_A + u^A_x + b^C_A u^A_x, a^A_x)$, $\omega^{CB}_x = (b^C_B b^B_x, b^C_B d^B_x, d^C_B + u^B_x + b^C_B u^B_x, a^B_x)$ from aggregators A and B, respectively. The effect of discounting in a transitive path is that uncertainty increases, not disbeliefs. Cluster C will then formulate a consensus $\omega^{AB}_x$ given aggregates from A and B, as well as the corresponding discounted beliefs $\omega^{CA}_x$ and $\omega^{CB}_x$. The consensus of two possibly conflicting opinions is an opinion that reflects both opinions in a fair and equal way (Figure 11). Again this can be accomplished by subjective logic. The effect of the consensus processing is to amplify belief and disbeliefs, and reduce uncertainty. The consensus result and an opinion will be forwarded towards the base station.
6. Simulation Evaluation

In this section, we report the results of simulation-based study on the effectiveness of our framework. The study is performed using QualNet network simulator [25]. We report a typical network setup for simulation in which a cluster consists of 25 nodes (Figure 12) with node 25 being the cluster head and is organized into two aggregation sets with nodes 1 and 13 being the aggregators, respectively. All but 4 nodes behave normally unless specified otherwise. Specifically, nodes 5, 15, and 16 misbehave all the time and node 6 misbehaves during time interval (150, 450) seconds. Normal nodes generate sensor readings, e.g., with a temperature at around 70F with certain variance while the sensor readings of misbehaving nodes may deviate from the norm. The simulation time is 32 minutes.

![An Example Logical Hierarchical Topology Used in QualNet Simulation](image)

6.1 Sensor Node Reputation Evolution

We first show the results of node reputation characterization and update. Figure 13 depicts the sensor node reputation evolution over time with two curves: one showing the reputation of a normal node and the other showing that of a misbehaving node. Clearly, after an initial warming
up period, a normal node quickly attains a high reputation (close to 1, the perfect reputation) and maintains a high reputation all the time. Misbehaving nodes, however, only achieve significantly lower reputation values. Node reputation values evolve based on actual sensor readings and their inherent statistical properties. The reputation value of a node at a particular instant reflects both the instantaneous reading and the past history of sensor readings.

Figure 14 shows a snapshot of the cluster-wide reputation of sensor nodes at the closing of the simulation. Note that node 6 misbehaves during (150, 450) seconds. Its reputation suffers when it misbehaves. Section 6.3 discusses more about a significant scenario that involves a cooperative malicious node such as node 6. Based on the reputation of nodes, misbehaving nodes can be readily identified and isolated.

![Figure 13. Sensor Node Reputation Evolution: a Normal Node Versus a Misbehaving Node](image-url)
6.2 Aggregation Result and Belief of Result with Misbehaving Nodes

One of the objectives of our framework is to enable trusted query and to quantify the extent of uncertainty in the returned response. This is achieved through a two-stage process in which aggregators obtain the first stage aggregation result and a quantification of uncertainty in terms of a belief value, and subsequently the cluster head fuses the aggregation results from multiple independent aggregation sets to provide much needed robustness.
Figure 16. Expected Belief Value at the Aggregator That Measures the Uncertainty in the Aggregate Sensor Reading

Figure 15 illustrates the aggregated sensor reading from an aggregation set while Figure 16 depicts the corresponding belief value of the aggregation result provided by the aggregator. Despite the existence of misbehaving nodes, the aggregation result appears to be immune to the impact of misbehaving nodes. Moreover, the aggregator expresses high confidence in its aggregation result as demonstrated by the expected belief values over time (Figure 16).

Figure 17. Aggregate Sensor Readings at the Cluster Head.
Figure 18. Expected Belief Value at the Cluster Head That Measures the Uncertainty in the Query Response

Similarly, by integrating inputs (both aggregation results and belief values) from two independent aggregation sets, Figure 17 and Figure 18 clearly show that the returned query response is immune to misbehaving nodes. More importantly, the user is offered a quantitative expression of how trustworthy the returned response is in the form of an expected belief value that accompanies the response.

6.3 Impact of Cooperative Malicious Node

We look into a scenario in which node 6 misbehaves more intelligently. The compromised node 6 first functions as a legitimate one till 150 seconds so that it can build up its reputation as high as other normal nodes. Later on from 150-450 seconds, it misbehaves and goes back to normal after 450 seconds. Figure 19 captures the reputation of node 6 as characterized by our mechanism. As seen from the figure, the reputation of node 6 suffers significantly and then gradually but slowly recovers after node 6 behaves normally. This cooperative malicious behavior is detected by our scheme. Therefore, its sensor readings are isolated to keep the aggregation result consistent with the true value all the time as shown in Figures 12-15.
Wireless sensor networks might be deployed in a malicious environment where it is very likely to be opposed to node compromise. Trustworthiness of individual sensor nodes can be characterized by using different techniques. In this chapter, we proposed a novel framework that exploits statistical theory as well as belief theory in order to achieve such characterization. The resulting technique was able to detect malicious nodes as well as misbehaving nodes. In addition to node compromise detection, a query to retrieve the aggregate response is highly trusted by looking at the expected belief of that response. The proposed framework is able to filter out any untrustworthy data and return the most-trusted aggregate response. We finally conducted a simulation based study to measure the effectiveness of the proposed framework.

Figure 19. Sensor Node Reputation Evolution: a Cooperative Malicious Node (Node 6)

7. Summary
V. NETWORK SECURITY VISUALIZER “SECVIZER”

In previous chapters we introduced the remarkable challenge of node compromise in sensor networks. Whenever the sensor node is being appropriated by an insider attack, any favorable attack can be launched. Several common attacks were summarized in the first chapter. One way of detecting and eliminating such attacks was to employ a trusted querying system (Chapter IV). Another way is to observe or scan the network for its traffic. Scanning though may generate huge data files. A more intuitive approach is to present the traffic in a visual view other than raw texts so as to ease the attack detection. In this chapter we present our developed network security visualization tool “SecVizer” which utilizes the parallel plot visualization technique and other visualization approaches to detect the various security vulnerabilities over a given network. Primarily, the tool is intended to visualize traffic generated by a network simulator. Visualization of and attack detection in captured real network traffic will be our future work.

1. Introduction and Related Work

Security in computer networks has been the subject of much research in the past two decades. Networks are vulnerable to various types of attacks such as port scanning, distributed denial-of-service (DDoS), and wormhole attack that targets wireless networks. Researchers have developed ways of detecting such vulnerabilities and act responsively, such as a TCP port scan detection tool called scanlogd [76].

One of the more recent methods of detecting vulnerabilities is through network traffic visualization. Visualization has been deployed in different fields and recently in visualizing network data. Visualization can be a powerful and effective technique if implemented carefully. Visualization can be very helpful in analyzing and understanding network simulations as well; therefore, recent network simulators (OPNET [64], QualNet [25]) have begun to employ visual
views to represent the network topology layout as well as packet level animation. These simulators usually generate huge files containing network traffic data and topology layout information which if presented visually can relieve the human eye and ease the task of network analysis from the user perspective. For Network Simulator-2 (NS-2) [65], an open source visualization tool called the Network Animator (Nam) was released and combined with the simulator. This tool is capable of drawing the simulated network topology in a two dimensional layout and can view network simulation traces as well as real world packet traces. The interactive NS-2 protocol and environment confirmation tool (iNSpect) [57], on the other hand, is an open source C++ OpenGL-based visualization tool that was created originally for visualizing and analyzing wireless networks simulated by NS-2 as Nam could not accomplish the mission (see Figure 20). The iNSpect tool also provides graphical user interface (GUI) which was designed using GTK+ [66]. Although these tools (Nam, iNSpect) add to NS-2 a new aspect of simulation analysis, we believe that they lack the specialized techniques in visualizing security data and thus detecting the different types of security attacks.

In the field of security data visualization, many tools including AfterGlow [71], EtherApe, GPL Cube of Potential Doom, NVisionIP, VisFlowConnect and recently rumint [69] have been developed and delivered to the public. rumint uses techniques like text rainfall, byte frequency, parallel coordinate plot [62], binary rainfall and scatter plot to visualize real-time traffic. Figure 21 illustrates the parallel plot used in rumint. Simulated traffic, on the other hand, is not considered by any of the mentioned tools.

The parallel coordinate plot technique was first invented in 1885 [61] and re-discovered in 1957 and since then has been an extremely powerful technique for visualizing multi-dimensional data including network data [62]. The technique is applicable to a diverse set of
multidimensional data where each dimension corresponds to an axis. $N$ axes are organized as uniformly spaced vertical lines. A data element in $N$-dimensional space manifests itself as a connected set of points, one on each axis. Compared to the traditional two-dimensional (x, y) and three-dimensional (x, y, z) plots with two and three variable axes respectively, the parallel coordinate plot can effectively show up to 25 axes at the same time despite the fact that it suffers from the noticeable occlusion problem (overlapping of both the line segments and labels). Picviz [70] is a popular parallel coordinates plotter that can be used to visualize various data inputs such as tcpdump, syslog, iptables log and more for fast discovery of interesting results.

Figure 20. iNSpect Simulation Visualization
We present our security visualization tool “SecVizer” (a quick and clear video demonstration can be viewed at [http://bmw.cs.wright.edu/projects/secviz](http://bmw.cs.wright.edu/projects/secviz)) which combines both topology visualization in a 3-dimensional perspective and the parallel coordinate plot technique used by rumint to obtain a faster and more effective detection of network vulnerabilities and thus leading to an early detection of the different security attacks. By simply observing image patterns of the parallel coordinate plot, one can conclude a malicious activity while at the same time exploring the network traffic volume and the topology being deployed.

In the following section we showcase the system’s architecture. Section 3 provides an overview of how the tool works by explaining the components of the graphical user interface (GUI). Section 4 studies simulation scenarios of several network security attacks and shows the effectiveness of using “SecVizer” in detecting such attacks. We conclude the chapter in Section 5.

### 2. SecVizer Architecture and Overview

The system’s architecture is illustrated in Figure 22. QualNet is used to create the desired simulation scenario. When creating the scenario, QualNet generates different files for different purposes with the necessary files being the (.nodes) file for node positioning, the (.config) file for configuring simulation parameters and the (.app) file which contains the different network
applications like CBR (Constant Bit Rate), FTP (File Transfer Protocol) and the super applications. The latter is a general purpose application-layer protocol provided by the QualNet simulator that can choose either reliable or unreliable transfer as its data delivery type. When placing a node on the canvas, QualNet scenario designer modifies the (.nodes) file and adds a new line describing the node’s position. Figure 23 illustrates an example of a (.nodes) file with 5 nodes being placed at different positions on the canvas.

Table 4 shows the description of each field in the first line of that file.

Table 4 shows the description of each field in the first line of that file.

![Figure 22. SeeVizer System Architecture](image)

After setting up the scenario on the canvas (placing nodes, adding applications, configuring traffic tracing, etc.), we run the simulation to obtain the traffic trace file (.trace). Trace files in QualNet are XML-based. Figure 24 shows one record of the trace while Table 5 only provides a description of the fields’ information that is being collected by our tool. QualNet
architecture is similar to that of the Open Systems Interconnection (OSI) model and is supported by a discrete-event engine. Each time an event is thrown in one layer in the hierarchy, a record is being saved in the trace file by the corresponding tracing protocol. The tracing protocol ids are provided at the beginning of the trace file as an identification map (e.g. the tracing protocol id ‘3’ in Table 5 maps to IPv4 protocol). We start by looking at the action code (an action code map is provided in Table 6); the record in Figure 24 represents an attempt of sending a 512-byte packet by node 10 having an IP address 192.0.0.10 at simulation time 1 second using UDP port number 5000 to the node with the IP address 192.0.0.1 on UDP port number 5212. To find the packet length, we subtract both the size of the UDP packet header (8 bytes) and the IP packet header size (20 bytes) from the total packet length (540 bytes).

![Figure 23. QualNet Nodes Positioning File Structure (.nodes)](image)

![Table 4. Description of the First Line Fields of Figure 23](table)

```
1 1 0 (1056.19, 1121.56, 0.0) 0 0
2 0 0 (805.05, 1173.17, 0.0) 0 0
3 0 (571.1, 1128.44, 0.0) 0 0
4 0 (326.83, 877.29, 0.0) 0 0
5 0 (715.6, 780.96, 0.0) 0 0
```

![Figure 24. QualNet Traffic Trace File (.trace) Format](figure)
Table 5. Description of Figure 24 Trace Record

<table>
<thead>
<tr>
<th>Field Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>originating node id</td>
</tr>
<tr>
<td>1.000000000</td>
<td>simulation time</td>
</tr>
<tr>
<td>10</td>
<td>processing node id</td>
</tr>
<tr>
<td>3</td>
<td>tracing protocol id</td>
</tr>
<tr>
<td>1</td>
<td>action code</td>
</tr>
<tr>
<td>5000</td>
<td>source port</td>
</tr>
<tr>
<td>5212</td>
<td>destination port</td>
</tr>
<tr>
<td>540</td>
<td>total length</td>
</tr>
<tr>
<td>192.0.0.10</td>
<td>source IP</td>
</tr>
<tr>
<td>192.0.0.1</td>
<td>destination IP</td>
</tr>
</tbody>
</table>

Table 6. Action Code Map

<table>
<thead>
<tr>
<th>Action code</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Send</td>
</tr>
<tr>
<td>2</td>
<td>Receive</td>
</tr>
<tr>
<td>3</td>
<td>Drop</td>
</tr>
<tr>
<td>4</td>
<td>Enqueued</td>
</tr>
<tr>
<td>5</td>
<td>Dequeued</td>
</tr>
</tbody>
</table>

Both files (.nodes, .trace) now serve as inputs to the proposed visualization tool. The tool parses the .nodes file and processes it in order to create a graphical view of the network topology while it parses and pre-processes the .trace file to set up the simulation playback and be able to visualize the network traffic.

The parsing process of the .nodes file is straightforward; however, parsing of the traffic trace is more complicated and needs much more attention. Figure 25 shows the parsing process of the QualNet trace file. The SecVizer parser starts by filtering out all records that were written by any tracing protocol other than IPv4. For example, if the routing protocol is AODV (Ad-hoc On-Demand Distance Vector), the parser will filter out all routing records because of their insignificance in the visualization process. It merely extracts the IPv4 records. The parser also filters out all the records that are related to a dropped or a queued packet, the visualization tool...
only shows those packets that are being sent or received. In the next section, we show the exact steps on how to use “SecVizer”.

Figure 25. Flowchart of the Process of Parsing the QualNet Trace File
3. Features of the Graphical User Interface

3.1 SecVizer Look and Feel

SecVizer is a platform-independent Qt-based [67] visualization tool that relies on OpenGL [63], one of the most powerful graphics rendering libraries, and the extended GLUT (OpenGL Utility Toolkit) [68] library for graphics rendering. It uses Qt together with C++, the main programming language of the Qt application development framework, to create the GUI shown in Figure 26.

The GUI consists of the playback control unit, view control unit, SecVizer configurations, parallel coordinate plot configurations, network topology window (top-right) and the parallel coordinate plot window (bottom-right). It also uses the external Qwt library [80] to create the nodes statistics window. Initially, all controls are disabled waiting for the user to load the topology file generated by QualNet (the .nodes file). After loading the .nodes file (by simply clicking the open icon), the topology is rendered in 3D orthographic-projected view inside the topology window. Now the user can use the controls under the “view control unit” for translation purposes (rotation around the 3 axes, zooming). In addition, the load trace button will be enabled. Whenever the trace is loaded, the simulation can be played, paused, or stopped using the controls under the “playback control unit”. Simulation progress is tracked by both the percentage progress bar and the simulation progress slider. The simulation speed can also be controlled under the same unit. The user has the choice of setting up some configurable parameters while running the simulation. The next subsection discusses such parameters.

3.2 SecVizer Current Features

1) 3D Rendering of Network Topology: Our tool can load a QualNet topology file (.nodes file) and displays it in 3D using OpenGL. Glut is being used to render a sphere representing a typical sending or forwarding node, whereas a tea pot is rendered representing the node whose IP
address is the intended destination address of the IPv4 packet. The intended destination address is the IP address of final destination (the destination IP in Figure 24).

![SecVizer Graphical User Interface (GUI)](image)

Figure 26. SecVizer Graphical User Interface (GUI)

2) **Parallel Coordinate Plot (PCP):** The user is able to choose up to seven different axes (see Figure 27) to be shown inside the PCP Window. The color scheme is as follows: packet sending intent is represented with a green line whereas a red line is being drawn whenever a packet is being received. Both TCP (Transmission Control Protocol) and UDP are processed
during the parsing process of the trace file; therefore, the user can choose to view both types of packets at the PCP at the same time represented by different axes.

3) **3D Transformations:** To rotate the entire topology around the 3 axes (x, y, z), the user can use the rotation sliders provided under the view control unit (see Figure 26). The user can also zoom in or out either by using the zoom slider under the view control unit or by rotating the mouse wheel and lastly, he/she can translate the entire topology by clicking the mouse left button and moving the mouse cursor to the desired location.

4) **Display of Node Labels:** The tool gives the choice of displaying the node ids (View -> Display -> Node Ids). It can also display the IP addresses of the nodes at the topology window. The mapping of node ids to IP addresses is constructed when parsing the trace file; therefore the IP address check item will not show up at the display sub-menu until the trace is loaded by the user. The topology window only shows the IP addresses of those nodes that are the sources of the generated packets.

5) **Trace Visualization:** SecVizer is able to load any corresponding trace file generated by the QualNet simulator and visualize that trace at the topology window as follows. Sending nodes
are green-colored. Receiving nodes, however, are colored with red. A line is drawn from source to destination showing the transmission path as well.

6) Animation controls: The tool also provides a playback control unit (see Figure 26) to control the simulation animation process (playing, stopping, pausing and slowing down the simulation). A progress bar is provided which shows the percentage of simulation completion over time, and finally, a simulation slider shows the entire progress from the beginning till the end of simulation.

7) Visualization Configuration Control: We currently provide the user with two configurable parameters; the node radius (how big the node should look like) and the transmission line elimination period (when should the tool remove the old lines off the display and reset the nodes color). If the last parameter was set to 0, it will show each single transmission on the screen and then removes it right away. However, if it is set to several seconds, the user might be able to better observe the different traffic patterns where lines are drawn for a longer period of time before they are removed.

8) Nodes Statistics: The tool includes a window that presents the collected nodes statistics dynamically during simulation playback using a bar graph. Figure 28 illustrates a bar graph of the total number of IP packets being sent by every node in the network and a bar graph of the total number of received IP packets. Qwt library [80] has been used to create such graphs.
4. Study of Various Security Attack Scenarios

The task of detecting security breaches in real-time networks is intricate, therefore, simulation is being thought of as an alternative and more effective solution. In [77] [78] and [79], researchers have used OPNET to simulate security including DoS [77] and DDoS [79]. As our tool is targeting QualNet traces, we have used QualNet to conduct simulations on the forthcoming scenarios.

As was described in section 2, we setup the network using QualNet in the first place. We use the setup of Figure 29 for all upcoming security attacks scenarios. The network is a wireless sensor network with several sensor nodes being deployed with sensor node id 1 being the gateway. All sensor nodes are mounted on Ethernet boards for IP address assignments as well as ports handling. The network-level routing policy is set to AODV routing protocol. In all the
scenarios, the parallel coordinate plot is pre-configured to show 4 axes (see Figure 26); source IP, destination IP, UDP destination port and packet length.

Figure 29. QualNet Topology Layout

4.1 Detection of DDoS attacks

Denial of Service (DoS) attack is a very common security attack where the attacker attempts to make the victim unable to provide its intended service to its users by preventing access to the target resource. Researchers have distinguished between two types of DoS attacks [72]: the ‘flood attack’ in which a continuous flood of traffic designed to consume resources at the targeted server (CPU cycles and memory) and/or in the network (bandwidth and packet buffers) can overwhelm the remote system, and the ‘software attack’ where several known software bugs on the target system (victim) can be exploited by a small number of malformed packets. The latter attack is easier to prevent by simply installing software patches to eliminate vulnerabilities or by setting up firewall rules that filter out the malformed packets. However, the flood attack is much harder to prevent/detect and many detection techniques [73] have been proposed to prevent or mitigate this type of attack.
DDoS is the distributed version of the flooded DoS attack. Since bandwidth consumption is the goal of a flooding DoS attack, the more bandwidth the attacker is able to work with, the more damage they can do. In a DDoS attack, the attacker first compromises a number of other hosts and installs daemons on them. Systems installed with such software are commonly referred to as bots and make up what is known as a botnet. These bots wait patiently until the attacker picks a victim and decides to attack. The attacker uses some sort of a controlling program, and all of the bots simultaneously attack the victim with some form of flooding DoS attack. Not only does the great number of distributed hosts multiply the effect of the flooding, this also makes tracing the attack source much more difficult. Although detecting DDoS attacks has been the subject of a few researches, not much of specialized work has been delivered yet. One not-fully-developed tool that we know of is called ‘Panoptis’ [74].

In this study, we have set up two simple DDoS scenarios at which bunch of sensor nodes were considered compromised by the attacker and were used to launch the attack against the targeted gateway (see Figure 29). In the first scenario, the attacker is targeting one single port at the gateway (192.0.0.1) while it targets multiple ports in the second scenario. In order to flood the gateway with several packets, we have used the QualNet super application with the delivery type being set to ‘Unreliable’ meaning that the packets are being delivered to UDP ports. After simulating the scenarios, the two generated files (.nodes, .trace) are fed into our tool to test its effectiveness. It has been found that our tool were able to visualize such scenarios and provide an early detection of the attacks by exploring noticeable traffic patterns at both the topology window and the parallel plot window. Figure 30 (a) shows the case where all compromised nodes have targeted the victim node at the single UDP port number ‘1025’, Figure 30 (b) shows the case where all compromised nodes have targeted the victim node at multiple UDP ports. In
both figures, all falsified packets had the same size of 512 bytes (see section 2). By investigating the parallel coordinate plot of such attacks, we can clearly identify the DDoS attack signature. The attacker has compromised number of sensor nodes and is flooding the targeted gateway (see the second left axis in Figure 30 (a) and (b)) with a number of falsified packets.

Figure 31 (a) and (b), on the other hand, are snapshots of the topology window when the attack was captured.

Figure 30. SecVizer Parallel Coordinate Plots of Different Simulated Security Scenarios.
4.2 Port Scan Detection

Port scanning (i.e. discovering hosts’ weaknesses by probing open ports [58]) is a very common technique that hackers rely on when they decide to attack a network. The attacker basically sends a number of packets to the victim and observes the response to find out what services are vulnerable on that target host.

There are several ways of scanning the target host for open ports. TCP scanners, for instance, use the ‘connect()’ network function of the operating system to connect to the target, if a port is open
the operating system completes the TCP three-way handshake and instantly terminates the connection. TCP scanning does not require special privileges and is clearly the simplest scanning technique. However, it is not used so often because it does not provide low-level control. Most commonly used TCP scanners are the SYN scanners; instead of using the operating system functions, they generate raw IP packets, send them to the target host and wait for responses. UDP scanners, on the other hand, generate UDP packets and send them to the target. If the port was not open, the host will reply with a port unreachable message. If it is open, there would be no response at all and the attacker can exploit this fact. Some computer systems prevent this attack by simply installing a firewall. The firewall will discard all port unreachable messages and block the port in order to mislead the attacker and make him think that the port is open. To pass through the firewall the attacker can use UDP scanners that send application-specific packets and wait for a response at the application layer.

Many scanning tools have been developed and are used to apply any of the above scanning techniques. One such tool is called Nmap [75]. We have mentioned that a firewall can prevent such scans, but what about if the target host is willing to detect such attacks and act upon. To achieve this, port scan detection tools were developed as well. For example, scanlogd [76] can be used for TCP port scan detection. Conti’s tool [60] can also provide a powerful mechanism of detecting the different types of port scans. In this study, we show how our tool can effectively detect port scans in its early stages. Figure 30 (c) illustrates the parallel coordinate plot of the attack scenario. Node 10 is probing node 1 for a number of UDP ports (axis 3 form left). Again, all probing packets have the same size of 512 bytes. By investigating the plot, we can clearly identify the port scan attack signature. Figure 31 (c) is a snapshot of the topology window when the attack was captured.
4.3 Host Scan Detection

Internet worms which represent a self-propagating malicious code rely intensively on host scan techniques to spread themselves. Host scans detect vulnerable machines in the network. Once vulnerabilities are detected, the previously infected machine propagates the worm code to vulnerable targets. Figure 30 (d) illustrates the parallel coordinate plot of the attack scenario; node 5 scans the network for vulnerable nodes. The infected node (node 5) sends random UDP packets to random destinations on port number 1025. By investigating the plot, we can clearly identify an attack signature of a possible host scan. Figure 31 (d) is a snapshot of the topology window when the attack was captured.

4.4 Nodes Statistics

For the DDoS attack scenarios the collected nodes statistics is shown in Figure 28. The total number of packets sent and received by each node in both scenarios is similar since the distributed compromised nodes are sending the same number of packets but setting the port number to a different number in the DDoS multi-port scenario while keeping the same port number in the DDoS single-port scenario. It is important here to mention that the total number of packets sent includes both the initially destined packets (i.e. the packets being generated by the sending nodes) and the packets being forwarded by the node. The tool does not distinguish between the forwarded and the initially generated packets at this time, a concern that will be considered for future work.
Figure 32 illustrates that just node number 10 is attacking the network by sending number of port scans (around 680 scanning packets) to one and only one victim which is node 1 (packets are only received by node 1 in Figure 32). Finally, Figure 33 shows the scenario where the host scan attack is being launched. Node 5 is flooding the network with a huge number of host scan packets (around 1500 scanning packets) and a number of hosts are receiving different number of scanning packet.
5. SecVizer Implementation Aspects

In this section we discuss how and what has been used to implement some of the tool features and primary functions.

1) Trace File Parsing: As discussed before the QualNet trace file has an XML-like format. Qt provides several ways to parse XML files such as using DOM (Document Object Model), SAX (Simple API for XML) and by using the QXmlStreamReader class. In our implementation, we have used QXmlStreamReader which comes with the new version of Qt (4.4) as being the fastest and most appropriate to use in the parsing procedure.

2) 3D Text Rendering: In order to render 3D text, an external library such as FTGL, Cairo and other platform dependent external libraries can be used. However in our implementation, we used the build-in Qt 4.4.3 QGLWidget::renderText() function. Using this function, an efficient
rendering of text, such as node labels, is achieved and the text is being transformed (rotated, zoomed, and translated) along with the topology.

3) **Qt, OpenGL and Glut integration:** The tool makes use of the three different libraries. All were integrated together to render the topology inside a Qt window. Qt was used to construct the GUI. An OpenGL Widget embedded inside the GUI was used for rendering the topology (here we can setup transformations, coloring, etc.) and finally in order to render nodes in 3D, we chose the GLUT library to render a sphere of a certain radius for each node. This is an advantage over iNSpect which renders circles and sticks with 2D.

4) **Animation:** The tool uses both the QTimeLine and the QTimer provided by Qt to deal with the animation part of the project. The QTimeLine manipulates simulation animation (coloring, drawing lines, etc.) while the QTimer is being set to delete transmission lines at a constant time interval.

5) **Nodes Statistics Bar Graphs:** Several libraries have been developed and employed for drawing business charts and technical applications. Some of those include KD Chart [81], plplot and Qwt [80]. Qwt, however, is an open source and provides an efficient histograms or bar graphs painting using Qt painters. The tool uses Qwt as an extended library to dynamically draw the nodes statistics graphs. Table 7 provides a summary of the libraries used in the development process of SecVizer.

<table>
<thead>
<tr>
<th>Library</th>
<th>Version</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qt</td>
<td>4.4.3</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>Glut</td>
<td>3.7.6</td>
<td>3-D Objects Rendering</td>
</tr>
<tr>
<td>Qwt</td>
<td>5.2.0</td>
<td>Nodes Statistics Bar Graphs</td>
</tr>
</tbody>
</table>

The sequence of the main actions that the user takes in order to observe the resulted topology view as well as the simulation playback animations begins by first loading the topology
file (.nodes), then loading the trace file and finally playing the simulation. Those three actions and the relative actions that are generated accordingly are shown in the sequence diagram of Figure 34. The sequence diagram basically describes the interactions among the user and the tool windows (windowing classes). The code implementation of some of the interaction functions is provided in Appendix A.

The primary classes that have contributed to the tool functionality are shown in the class diagram of Figure 35. Both the class diagram and the sequence diagram can be viewed online at (http://bmw.cs.wright.edu/projects/secvizer/docs/class_diagram/). The code documentation of the tool can also be found at (http://bmw.cs.wright.edu/projects/secvizer/docs/).

![Figure 34. SecVizer Sequence Diagram Illustrating the Interactions among the Different Windows](image_url)
Figure 35. SecVizer Class Diagram
6. Summary

In this chapter, we have presented “SecVizer”, a novel network security visualization tool that can detect different types of security attacks by exploiting the visual recognition of the human eye. It can render any QualNet topology in a 3 dimensional perspective. It can play animation of any QualNet-generated traffic trace as well. It has been demonstrated that the parallel coordinate plot used in our tool serves as an effective visualization technique toward detecting different security attacks. We have shown scenarios where it was able to detect DDoS attacks, port scanning as well as host scans.
VI. CONCLUSION AND FUTURE WORK

In Chapter IV, we showcased a simulation based study of our proposed trust-based scheme for wireless sensor networks. We are planning to further test the effectiveness of the scheme through hardware implementation over xbow motes. The testing will also involve storing the belief values (levels of trust) received by the base station together with the sensed value into a LAN database server in order to further reason over or make decisions upon the trusted readings. Different software components such as TinyOS, necS and TinyDB will be used.

As being the initial version of the SecVizer, we were only applying the parsing process on QualNet traces. However, our plan is to extend the parser to include other types of traces generated by different network simulators such as OPNET and NS-2. We also plan to capture real time traffic instead, and deal with traffic captured by packet sniffers such as Wireshark.

Our SecVizer tool does not distinguish between wired and wireless networks. Our intention, however, is to achieve such distinction and support mobility visualization for wireless networks. Nodes statistics can also be improved by collecting the number of dropped, enqueued and dequeued packets. By doing this, additional security attacks like replay attacks can be detected by observing the bar graph of the unexpectedly enqueued packets. Further, a distinction between the forwarded packets and the initially destined packets will improve the task of attack detection. Furthermore, to overcome the occlusion problem of the parallel coordinate plot, our plan is to provide a mechanism of window docking. The user can simply undock the topology or the parallel coordinate plot window and maximizes it to his/her best vision.

Last but not least, despite the fact that the SecVizer tool provides an option for disabling graphics rendering at the unwanted windows, it is still our plan to improve the rendering performance by using advanced OpenGL techniques.
void SecVizerTopologyWindow::loadTopologySlot(QString topologyFileName)
{
    this->nodesList.clear();  // remove old nodes from topology
    this->reset();

    QFile topologyFile(topologyFileName);
    if (!topologyFile.open(QFile::ReadOnly | QFile::Text)) {
        QMessageBox::warning(this, tr("Application"),
                               tr("Cannot read file %1:
                                   %2.")
                               .arg(topologyFileName)
                               .arg(topologyFile.errorString()));
        return;
    }

    QTextStream in(&topologyFile);
    while (!in.atEnd())
    {
        QString input_line = in.readLine();
        QStringList tokens_list = input_line.split(' ');
        SecVizerNode *node = new SecVizerNode();

        /* parse each line in .nodes file for the node_id and the x,y,z coordinates */
        node->setNodeId(tokens_list.at(0).toInt());
        node->setPositionX((tokens_list.at(2).mid(1,tokens_list.at(2).size()-2)).toDouble());
        node->setPositionY((tokens_list.at(3).left(tokens_list.at(3).size()-1)).toDouble());
        node->setPositionZ((tokens_list.at(4).left(tokens_list.at(4).size()-1)).toDouble());
        this->nodesList.append(node);
    }

    if (DEBUG)
        this->printNodesList();

    this->topologyLoaded = true;
    this->updateOrtho();
    this->updateGL();

    // connect to the Parallel Plot Window initiation Slot "initializePlotWindow"
    this->connect(this, SIGNAL(initializePlotWindow(QVector<SecVizerNode *> *)),
                  this->parentWidget()->layout()->itemAt(1)->widget(),
                  SLOT(initializePlot(QVector<SecVizerNode *> *)));

    emit initializePlotWindow(&this->nodesList);
}
void SecVizerTopologyWindow::paintGL()
{
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);  // set background color to Black
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (OPENGL_DEBUG)
        printf("Ortho: %f,%f,%f,%f,%f,%f\n", this->topologyOrtho->getLeft(),
            this->topologyOrtho->getRight(),
            this->topologyOrtho->getBottom(),
            this->topologyOrtho->getTop(),
            this->topologyOrtho->getNear() - 3000,
            this->topologyOrtho->getFar() + 3000);
    glOrtho (this->topologyOrtho->getLeft(),
        this->topologyOrtho->getRight(),
        this->topologyOrtho->getBottom(),
        this->topologyOrtho->getTop(),
        this->topologyOrtho->getNear() - 3000,
        this->topologyOrtho->getFar() + 3000);
    this->drawAxes();
    this->moveOrigin();  //translate to the center of the topology for rotation purposes
    glScalef(this->scaleNetwork->getScale_x(), this->scaleNetwork->getScale_y(),
        this->scaleNetwork->getScale_z());  // scale the nodes
    glPushMatrix();
    glTranslatef(this->translateNetwork->getTranslate_x(), this->translateNetwork->
        getTranslate_y(), this->translateNetwork->getTranslate_z());

    // rotate around x
    glRotated(this->rotateX->getRotation_angle(),
        this->rotateX->getRotate_x(),
        this->rotateX->getRotate_y(),
        this->rotateX->getRotate_z());

    // rotate around y
    glRotated(this->rotateY->getRotation_angle(),
        this->rotateY->getRotate_x(),
        this->rotateY->getRotate_y(),
        this->rotateY->getRotate_z());

    // rotate around z
    glRotated(this->rotateZ->getRotation_angle(),
        this->rotateZ->getRotate_x(),
        this->rotateZ->getRotate_y(),
        this->rotateZ->getRotate_z());

    glColorMaterial(GL_FRONT_AND_BACK, GL_EMISSION);
    glEnable(GL_COLOR_MATERIAL);

    // draw the nodes
    this->drawTopology();
    this->drawActiveRecords();
    glPopMatrix();
}

Figure 37. The Main OpenGL Drawing Function under the Topology Window
void SecVizerTopologyWindow::drawTopology()
{
  glTranslatef(-1 * this->centerX, -1 * this->centerY, -1 * this->centerZ); // translate back to the origin (0,0,0)

  for(int i = 0; i < this->nodesList.size(); i++)
  {
    // set node's color
    glColor3f(this->nodesList.at(i)->getNodeColor()->red,
              this->nodesList.at(i)->getNodeColor()->green,
              this->nodesList.at(i)->getNodeColor()->blue);

    // translate the origin to the node position
    glTranslatef(this->nodesList.at(i)->getPositionX(),
                 this->nodesList.at(i)->getPositionY(),
                 this->nodesList.at(i)->getPositionZ());

    // draw the node
    glutSolidSphere(this->nodesList.at(i)->getNodeRadius(),1000,1000);

    if(this->displayNodeId) // if the display node id is checked
    {
      QString nodeIdStr;
      nodeIdStr.setNum(this->nodesList.at(i)->getNodeId());
      glColor3f(1.0,1.0,1.0);
      this->renderText(this->nodesList.at(i)->getNodeRadius(),
                        this->nodesList.at(i)->getNodeRadius(),
                        this->nodesList.at(i)->getNodeRadius(),
                        nodeIdStr);
    }

    if(this->displayIPs) // if the display IPs is checked
    {
      glColor3f(1.0,1.0,1.0);
      this->renderText(this->nodesList.at(i)->getNodeRadius(),
                        this->nodesList.at(i)->getNodeRadius(),
                        this->nodesList.at(i)->getNodeRadius(),
                        this->nodesList.at(i)->getNodeIPAddress());
    }

    // translate the origin back to (0,0,0)
    glTranslatef(-1 * this->nodesList.at(i)->getPositionX(),
                 -1 * this->nodesList.at(i)->getPositionY(),
                 -1 * this->nodesList.at(i)->getPositionZ());
  }
}

Figure 38. Code Implementation of the Topology Rendering Function
void SecVizerTopologyWindow::drawActiveRecords()
{
    GLuint linesList = glGenLists(1);
    glEndList(linesList, GL_COMPILE);
    for(int i = 0; i < this->activeRecordsList.size(); i++)
    {
        double line_coord_x_1 = this->nodesList.at(this->getNodeIndex(this->activeRecordsList.at(i)->getOriginatingNodeId()))->getPositionX();
        double line_coord_y_1 = this->nodesList.at(this->getNodeIndex(this->activeRecordsList.at(i)->getOriginatingNodeId()))->getPositionY();
        double line_coord_z_1 = this->nodesList.at(this->getNodeIndex(this->activeRecordsList.at(i)->getOriginatingNodeId()))->getPositionZ();
        double line_coord_x_2 = this->nodesList.at(this->getNodeIndex(this->activeRecordsList.at(i)->getProcessingNodeId()))->getPositionX();
        double line_coord_y_2 = this->nodesList.at(this->getNodeIndex(this->activeRecordsList.at(i)->getProcessingNodeId()))->getPositionY();
        double line_coord_z_2 = this->nodesList.at(this->getNodeIndex(this->activeRecordsList.at(i)->getProcessingNodeId()))->getPositionZ();

        if(this->topologyLoaded)
        {
            glColor3f(1.0,1.0,1.0);
            glLineWidth(1);
            glBegin(GL_LINES);
            glVertex3f(line_coord_x_1, line_coord_y_1, line_coord_z_1);
            glVertex3f(line_coord_x_2, line_coord_y_2, line_coord_z_2);
            glEnd();
        }
    }
    glEndList();
    glCallList(linesList);
}

Figure 39. Code Implementation of the Active Records Rendering Function
REFERENCES


64. OPNET— a leading provider of solutions for managing networks and applications. (http://www.opnet.com/).

65. The network simulator - NS-2. (http://isi.edu/nsnam/ns/).

66. GTK+ is a highly usable, feature rich toolkit for creating graphical user interfaces which boasts cross platform compatibility and an easy to use API. (http://www.gtk.org).

67. Qt is a cross-platform application framework. (http://www.qtsoftware.com/products).

68. GLUT - The OpenGL Utility Toolkit. (http://www.opengl.org/resources/libraries/glut/)

69. rumint (room-int)— an open source network and security visualization tool (http://www.rumint.org/).
70. Picviz— a parallel coordinates plotter which enables easy scripting from various input (tcpdump, syslog, iptables logs, apache logs, etc..) to visualize your data and discover interesting results quickly (http://www.wallinfire.net/picviz).

71. AfterGlow— a collection of scripts which facilitate the process of generating graphs (http://afterglow.sourceforge.net/).


76. Scanlogd— a port scan detection tool (http://www.openwall.com/scanlogd/).


82. InetVis— a 3-D scatter-plot visualization for network traffic. In way, it's more or less like a media player, but for network traffic. It's quite handy for observing scan activity and other traffic patterns. (http://www.cs.ru.ac.za/research/g02v2468/inetvis.html). Page accessed on March 28, 2009.


85. Wireshark— an award-winning network protocol analyzer developed by an international team of experts. (http://www.wireshark.org/).

86. Wine— run Windows applications on Linux, BSD and MAC OS X. (http://www.winehq.org/).
VITA

Giovani Rimon Abuaitah was born in Bethlehem, Palestine on the 15th of December 1983. Throughout his childhood, he attended the Evangelical Lutheran School in Beitsahour, his hometown, and graduated from there in June 2001. Right after he graduated from high school, he enrolled in the five-year computer systems engineering program that was offered for the first time at Birzeit University located in the West Bank. He graduated from there in June 2006 with a Bachelor of Science degree. In September 2007, he enrolled in the Master of Science in computer engineering (MSCE) program at Wright State University located in Dayton, Ohio, USA. Since January 2008, he has been working as a graduate research assistant at the Broadband, Mobile and Wireless Networking Research Laboratory (BMW Lab) towards his Master of Science degree for which this thesis is a partial fulfillment.