AUTHENTICATED MESSAGING IN WIRELESS SENSOR NETWORKS USED FOR SURVEILLANCE

by

Raymond Sbrusch, B.A.

THESIS

Presented to the Faculty of

The University of Houston Clear Lake

In Partial Fulfillment

of the Requirements

for the Degree

MASTER OF SCIENCE

THE UNIVERSITY OF HOUSTON-CLEAR LAKE

May, 2008

AUTHENTICATED MESSAGING IN WIRELESS SENSOR NETWORKS USED FOR SURVEILLANCE

by

Raymond Sbrusch

APPROVED BY

T. Andrew Yang, Ph.D., Chair

Gary Boetticher, Ph.D., Committee Member

Alfredo Perez-Davila, Ph.D., Committee Member

Dennis M. Casserly, Ph.D., CIH, Associate Dean

Sadegh Davari, Ph.D., Dean

DEDICATIONS

I dedicate this to my parents, who provided unending encouragement and the best geek toys of the early 1980s. They have enabled me to turn a childhood hobby into a rewarding pursuit.

ACKNOWLEDGMENTS

I sincerely thank my thesis committee chair, Dr. T. Andrew Yang, for his tireless guidance throughout my information security education. I have grown from a security technician into a well-rounded professional with his support. I also acknowledge the support of my thesis committee. Dr. Alfredo Perez-Davila forced me to drop iterative logic and reason about relationships among objects. Dr. Gary Boetticher inspired me to further abstract concepts into higher domains. I truly appreciate the detail Dr. Boetticher puts into providing an incomparable educational experience for his students.

I would also like to acknowledge Dr. Ernst Leiss, from the University of Houston, for the crash course in automata and computability. He put the “science” in Computer Science.

Finally, I thank my friends and family, especially Daranee, for their never-ending patience.

ABSTRACT

AUTHENTICATED MESSAGING IN WIRELESS SENSOR

NETWORKS USED FOR SURVEILLANCE

Raymond Sbrusch, M.S.
The University of Houston, Clear Lake, 2008

Thesis Chair: T. Andrew Yang, Ph.D.

Wireless sensor networks simplify the collection and analysis of data from multiple locations. The self-organization capabilities of wireless sensor networks enable rapid deployment of target tracking and intrusion detection applications in hostile environments. However, sensor networks deployed in adversarial environments must be fortified against attacks. This thesis examines the threats against wireless sensor networks and surveys countermeasures that protect their communications with origin integrity and data integrity. This thesis solves the security problem in wireless sensor networks deployed for surveillance and target tracking by application of appropriate security mechanisms to a target tracking method. This new target tracking method, called s(OCO), provides reasonable countermeasures that mitigate vulnerabilities identified in a formal risk assessment. A simulation of the new methods reveals the actual cost of securing a target tracking method.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

Wireless sensor networks simplify the collection and analysis of data from multiple locations. Target tracking and perimeter intrusion detection applications benefit from the ad-hoc deployment and self-organization capabilities of wireless sensor networks. However, sensor networks deployed in hostile environments must be fortified against attacks by adversaries. This thesis examines the constraints that make wireless sensor network surveillance challenging and evaluates algorithms that provide origin integrity and data integrity for wireless sensor networks. This thesis solves the security problem in wireless sensor networks deployed for surveillance and target tracking by applying appropriate security mechanisms to a target tracking method, Optimized Communication and Organization [1].

1.1 Proposal

This thesis develops an authentication mechanism to protect the Optimized Communication and Organization (OCO) method for routing and self-organization of a wireless sensor network [1]. This includes authentication of sent messages to assure that they have not been altered (aka. message integrity), and authentication of the sender to assure that the messages are not forged (aka. origin integrity). The process begins with a survey of security threats and risk mitigation strategies common to all wireless sensor networks. This survey includes mainly attacks against origin and message integrity, as well as those against confidentiality and availability. The risk analysis converges into a risk

assessment of OCO messaging, following the methodology outlined in [2]. Selection of an elegant authentication solution requires a survey of current unicast and broadcast message authentication protocols for wireless sensor networks. The protocols will be contrasted to select the most appropriate one for OCO. Following the model in [3], the goal is to find a small set of cryptographic primitives that can be used for all OCO messages. The thesis then demonstrates via simulation with the network simulation software OMNeT++ [4] the energy costs of integrating message authentication into OCO. Understanding this cost enables an application owner to evaluate whether to accept the risk of insecure messaging or bear the cost of authentication.

1.2 Benefits

Deployment of wireless sensor networks in critical military and civilian applications demands secure authentication. Without authentication mechanisms tailored to the application, sensor networks will be unreliable for use in critical arenas. The receiver must be guaranteed that critical messages indeed originated from the claimed source. They must also be able to confirm that a message was not altered in transit. Conventional security mechanisms in use on the Internet are usually not applicable to wireless sensor networks because of the limited resources available in the sensor nodes, such as limited processor speed, smaller memory size, and limited communication channels and speed. While security mechanisms have been proposed for wireless sensor networks, one cannot haphazardly apply a security protocol to a network without first understanding the functional and security requirements of the application. Security comes at a cost; and that cost must be balanced with the goals of the application. The goal of this research is to

produce an efficient, authenticated version of the OCO method, which effectively provides both message integrity and origin integrity to the wireless sensor network applications.

1.3 Thesis Outline

Applications of wireless sensor networks span across diverse fields from military surveillance across enemy lines to monitoring of avian breeding habits in sensitive wildlife habitats. Sensor networks simplify the simultaneous collection and organization of data from multiple locations, which may be unreachable, inhospitable, or even hostile environments. The wired counterparts of these sensors have been utilized in industrial and military applications for decades, but the simple constraint of stringing cables to the monitored site limits the reach of this communication mechanism. Merging wireless communications with sensor network capabilities enables rapid deployment and reduces the cost of the infrastructure. However, adopting wireless communications introduces a new set of challenges. Section 2 expands this analysis of wireless sensor networks used for target tracking, including examination of resource constraints and other challenges.

Section 3 follows the evolution of target tracking methodologies from conventional protocols of direct communication to tree topologies organized for network longevity. Target tracking techniques may focus on intruders penetrating a border in a two-dimensional space or may try to provide comprehensive coverage of a detection region. These techniques commonly impose a trade-off between accuracy of the tracking algorithm and efficient use of sensor energy, computation power, and communication resources. The naive surveillance strategy delivers high accuracy by enabling the sensing mechanism on every node in the network and forcing each node to send alerts directly to the base station.

This strategy reduces network longevity and increases contention for the radio channel. Advanced strategies organize nodes into clusters with child nodes detecting an intrusion and intermediary nodes relaying messages between children and the base station. Section 3 illustrates how methods, including Scalable Tracking Using Networked Sensors (STUN) [5], Low Energy Adaptive Clustering Hierarchy (LEACH) [6], and Optimized Communication and Organization (OCO) [1], and Guard Duty Alarming Technique (GDAT) [7] balance network longevity and detection accuracy.

Secure wireless sensor network deployments remain elusive because of resource constraints; however, operation of sensor networks in hostile locations requires secure, authenticated communication. Section 4 categorizes threats against wireless sensor networks and culminates with an OCO risk assessment, reviewing OCO messages in terms of their value to the attacker and their respective impact upon the reliability of the OCO network. Section 5 surveys authentication protocols that can be employed to mitigate threats against wireless sensor networks. The survey tracks the evolution of sensor network authentication from widely adopted standards including SPINS [3], and TinySec [8], to more comprehensive strategies. Section 5 closes comparing power utilization characteristics of the security standards.

Sections 6 and 7 formalize the problem statement and introduces the solution, s(OCO), a secured implementation of OCO featuring message integrity and origin integrity. This implementation strives to provide a sufficient amount of security without significantly reversing the efficiency improvements in OCO. The layer of security enveloping OCO,

derived from TinySec-Auth [8], employs cryptographic primitives with processor and memory utilization tuned for constrained sensor nodes.

Section 8 and Section 9 put forward an empirical evaluation of s(OCO) to appraise the impact of security on network longevity. This evaluation follows the methodology outlined in [1] by simulating node communications in the OMNeT++ network simulator. The simulation contrasts unauthenticated OCO with s(OCO) in experiments of a 500 node network and a 1000 node network. These experiments allow empirical assessment of the cost of authentication on network longevity.

2 Wireless Sensor Networks

The term wireless sensor network (WSN) describes an association among miniaturized embedded communication devices that monitor and analyze their surrounding environment. The network is composed of many tiny nodes sometimes referred to as motes [9]. A node is made up of the microcontroller, the sensor(s), the radio communication component, and a power source. Wireless sensor nodes range in size from a few millimeters to the size of a handheld computer. Regardless of size, sensor nodes share common constraints. This section identifies the unique challenges of wireless sensor networks and leads into a survey of organization, communication, and tracking algorithms.

2.1 Characteristics of Wireless Sensor Networks

Wireless sensor networks are deployed for a diverse variety of applications, each characterized by a unique set of requirements. The authors of [10] organized a workshop with the European Science Foundation to classify application domains, define hardware and software requirements for each domain, and determine how to coordinate research into wireless sensor networks. Their results counter the general conviction that the most common scope for research into wireless sensor networks centers on military applications, which involve large-scale ad-hoc networks with homogeneity among tiny, resource-constrained, immobile sensor nodes. The results uncover the inadequacy of this rigid definition. The participants in the workshop extract 14 characteristics of commercial and research networks. These characteristics consist of deployment, mobility, resources, cost,

energy, heterogeneity, mobility, infrastructure, topology, coverage, connectivity, size, lifetime, and quality of service. These attributes influence design decisions, especially those related to security. They will be employed to characterize target tracking wireless sensor networks in Section 3.

While the classical sensor network consisted of homogeneous devices, contemporary sensor networks incorporate modular design and make use of heterogeneous nodes that fulfill unique requirements. For example, some nodes include a GPS sensor that other nodes can query to determine their location. Others may include interfaces to the Internet through satellite or cellular communications. While radio frequency is the most common communication modality, information can also be transmitted via laser, sound, and diffuse light.

These communication capabilities support an assortment of network infrastructures. In a basic infrastructure-organized network, nodes can only communicate with a base station. The opposite is true in an ad-hoc network where there is no base station or communication infrastructure. In this case, each node can communicate with any other node. The communication infrastructure influences network topology. In some cases, each node must be within radio range of any other node because messages can only travel across a single hop. Networks organized into a graph-like topology allow routing of messages across multiple hops.

Some applications can achieve their goals with a network of sparsely deployed sensors. Others require a densely populated network with redundant nodes available.

Network topology and coverage requirements determine the network size. Networks may range in size from thousands of nodes to only a few. Target tracking networks such as OCO and GDAT have evolved to embrace densely populations as a means to improve efficiency while maintaining accuracy.

2.2 Challenges of Target Tracking

Target tracking wireless sensor networks face a unique set of challenges when compared to sensor networks deployed for other applications such as building automation or habitat monitoring. Threats against the network increase with the hostility of the surveilled area and range from simple node destruction to sophisticated network-based exploits. The sensor network must be able to monitor intrusions and securely deliver alerts even if a large percentage of the nodes have been compromised.

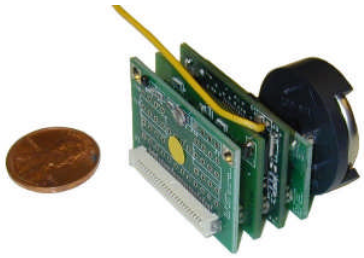

Figure 1: The Rene Mote [9]

2.2.1 Resource Constraints

Secure and reliable wireless deployments remain elusive because of four prohibitive constraints. Wireless sensor nodes, such as the Rene mote in Figure 1 [9], are characterized by limited computational abilities, small amounts of permanent and temporary storage, and finite energy resources. The wireless communication medium introduces reliability and security risks that, if left unresolved, preclude the use of wireless

sensor networks in adversarial locations. Conventional security mechanisms in use on the Internet are usually not applicable to wireless sensor networks because of the limited resources of the nodes. Sensor network packets commonly contain 32 bytes or less [11]. Conventional security mechanisms that add 16 bytes of network overhead are inappropriate for use in wireless sensor networks since they will quickly drain the sensor power source. On some platforms, each byte transmitted consumes about as much power as executing 4000-5000 instructions [8].

Besides the requirement of conserving energy, a wireless sensor network deployed for target tracking must provide a robust communication channel, real-time alerting, resistance to tampering and stealth. This paradox forces network designers to exploit ways to improve efficiency while maintaining a high level of accuracy by, for example, incorporating redundancy, clustering, etc. into the network.

2.2.2 Physical Threats

Deployment of a sensor network in a safe environment can be carefully planned and controlled. Safer environments can benefit from more deliberate node placement. For example, in the Twenty-nine Palms experiment [9], researchers at U.C. Berkeley deployed wireless sensor nodes to track the path of tanks along a predetermined path. A small number of sensors were carefully dropped from an unmanned aerial vehicle at points near the tank path. The sensors detected the presence of the tanks using a magnetometer and sent alerts over a 916.5 MHz radio signal to the base station. This application detected the arrival of the tank and calculated its speed and direction as it moved along the path.

In contrast, the challenges commence as soon as sensor nodes are deployed in a hostile environment, where they become subject to numerous threats. An adversary can physically destroy or displace nodes or launch network attacks against node communications. External forces such as explosions or earthquakes can unpredictably relocate nodes. Aerial dispersion results in a highly random coverage pattern with coverage gaps in some areas and excessive redundancy in others. Once sensors are deployed in an adversary's domain, there are few chances to modify the coverage patterns or refresh dying nodes.

3 Target Tracking Methods

Detection and tracking of objects moving through a surveilled region remains in the forefront of wireless sensor network research. In wireless sensor networks deployed for tracking objects, the effectiveness of organization and communication methods depends on many factors, including the node deployment tactic, the layout of the surveyed landscape, and the path of the intruder. Careful selection of appropriate communication and organization protocols can assure longer life for the network. Target tracking strategies can be evaluated on how they organize to form a network, how they sense intruders, how they communicate results to the base station, and how they reorganize to handle exhausted or damaged nodes. This section surveys target tracking methods and shows the trend in target tracking research toward improving efficiency and network longevity.

3.1 Single-Hop Communication

The simplest organization and communication strategy, known as Direct Communication (DC) [6], requires each node to transmit alerts directly to the base station. This single-hop strategy suffers from rapid depletion of resources. In DC, each mote monitors the environment with its sensing module and transmits alerts directly to the base station over a wireless channel. While this delivers the most accurate detection of intrusions from any attack vector, DC faces a number of setbacks.

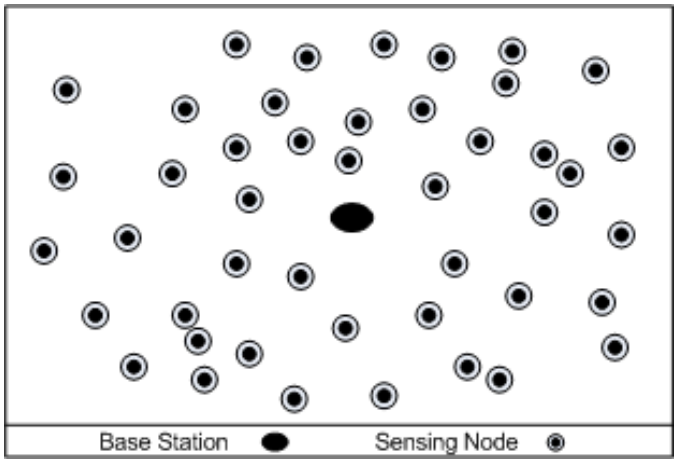

Figure 2: Direct Communication Network

Figure 2 illustrates a DC network composed of sensing nodes and a base station. Each sensor node must have a radio transmitter powerful enough to reach the base station. This limits the effective range of the network and forces the base station to allocate enough radio channels for communication with all sensors. Motes maintain an activated sensor module throughout their lifetime. This module continuously draws power from the node battery. Since nodes may have overlapping sensor coverage areas, redundant alerts may be sent to the base station, again unnecessarily depleting the battery. While theoretically effective at tracking targets, DC proves inefficient and impractical for use in real-world applications.

3.2 Hierarchy Trees

Organizing nodes into hierarchical network topologies or clusters can increase the coverage area and prune redundant alerts. In a tree-structured network organization, senor nodes occupy graph vertices and graph edges signify direct communication links between nodes. Scalable Tracking Using Networked Sensors (STUN) [5], based on a hierarchy tree,

aims to track a large number of objects as they move through the surveilled region. STUN organizes sensing nodes into a linear graph; however, adjacency of nodes in the graph does not necessarily imply physical nearness. Leaf nodes at the bottom of the STUN tree function as sensing nodes. Nodes at intermediate levels relay messages from sensing nodes to the base station at the root of the tree. Intermediary nodes store information about the presence of detected objects. When leaf nodes send detection messages toward the base station, the intermediate nodes compare the alerts to the information they already recorded and drop the messages if they are redundant. Pruning redundant alerts lowers communication costs. While hierarchy trees are an improvement over Direct Communication, sensor networks gain more efficiency and accuracy by accounting for the physical proximity of the sensor nodes after they are dispersed.

3.3 Clustering Methods

Clustering methods capitalize on ability to detect node proximity when forming the hierarchy tree. Like a hierarchy tree, clustering algorithms prune redundant messages as they are sent to the base station. They also conserve energy by assuring low cost radio communication between sensor nodes and intermediate nodes. The Low Energy Adaptive Clustering Hierarchy (LEACH) [6] illustrates how knowledge of actual node proximity can improve efficiency of hierarchy tree based organizations. During the LEACH setup phase, nodes elect themselves to be local cluster heads and broadcast messages to their neighbors advertising their status.

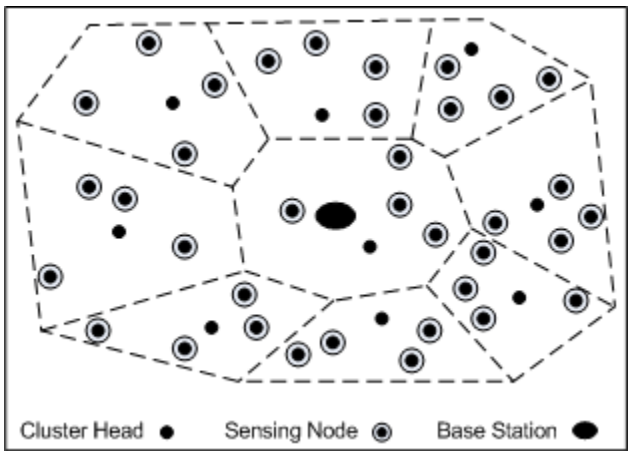

Figure 3: LEACH Network Organization

The cluster heads act as intermediate nodes and relay messages between a neighborhood of leaf nodes and the base station. To qualify for the role of cluster head, a node must have enough power to relay messages to the base station. Sensor nodes analyze cluster head advertisements to determine the cost of wireless communication with the cluster head. They choose which neighborhood they want to join by selecting the cluster head that requires the least radio transmission power. Figure 3 shows a LEACH network organized into nine neighborhoods. The role of cluster head consumes more energy than the role of sensor node, thus a LEACH network periodically repeats the setup process, electing new nodes to the cluster head position.

LEACH distributes the cost of serving as cluster head among all nodes in the network, thus increasing the lifetime of the network. Like STUN, LEACH still lacks knowledge of the surveilled terrain. Since any node could theoretically elect itself as the cluster head, LEACH requires that all nodes initially have enough radio power to communicate directly with the base, even if they are on the perimeter of the surveilled

region. LEACH also suffers from gaps in coverage because the cluster-head election process lacks knowledge of the surveilled region. The election process only evaluates the strength of the node's radio, not the coverage patterns of neighboring nodes. As with other proposals introduced until now, all nodes in the network activate their sensing component and their radio interface.

3.4 Energy-Efficient Clustering

Many wireless sensor network target-tracking approaches assume that an intruder will enter the detection region from the perimeter. These techniques conserve energy by allowing interior nodes to sleep until an intrusion occurs, thus increasing the longevity of the network. However, this strategy leaves the interior of the network vulnerable to attacks that bypass the perimeter, such as aerial or insider attacks. Rababaah and Shirkhodaie [7] propose a clustering and tracking technique that significantly increases efficiency comparable to basic clustering techniques while maintaining sensing throughout the detection region. This method, called Guard Duty Alarming Technique (GDAT), is inspired by military practice of rotating guard duty. While most soldiers sleep, one soldier is ordered to remain on guard duty for an assigned period. In GDAT, one sensing node is “on guard,” actively sensing while its other cluster members sleep.

GDAT requires two types of nodes: head nodes and sensing nodes. Head nodes are provisioned with both strong radios to communicate with the base station and with global positioning receivers to append location information to intrusion alerts. Sensing nodes are equipped with intrusion detection devices and only require sufficient power to communicate with the head nodes.

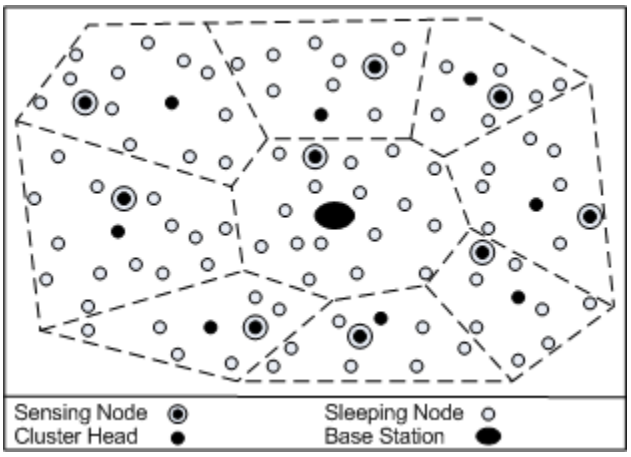

Figure 4: GDAT Organization

Following network deployment, both head nodes and sensing nodes are awake. Head nodes broadcast advertisements requesting that sensing nodes adopt them as their cluster head. Sensing nodes accept the first offer they receive, record the ID of the head node, and send an acceptance message to the head node. Each head node can accept 32 sensing nodes into its cluster.

The tracking phase begins once clustering is complete. The cluster head assigns one sensing node to “guard duty” for a small interval and instructs the other sensing nodes to sleep. The sensing node can reside anywhere within the cluster. When the sensing node detects an intrusion, it alerts the head node, which appends GPS coordinates to the alert and forwards it to the base station. Following the intrusion, the sensing node will remain on duty until its shift is complete. At that point, the head node will assign another sensing node to guard duty. Figure 4 highlights the advantage of GDAT over the LEACH network

depicted in Figure 3. In LEACH, all nodes drain energy with their activated sensors and radios. In GDAT, the majority of nodes sleep.

Important assumptions affect the success of GDAT. Most significantly, GDAT assumes that head nodes and sensors will be normally distributed across a detection region following a probability distribution function. Factors such as rugged terrain, battle, or natural forces can adversely influence the distribution of head nodes and sensing nodes. In comparison to image processing techniques, which will be described in the following subsection, GDAT leaves the perimeter porous. Even though it delivers higher accuracy throughout the network, it may not detect an intruder crossing the perimeter as quickly as an image processing technique. The authors indicate that they will include image-processing techniques in future GDAT research.

3.5 Image Processing Techniques

Image processing techniques have been shown to be a more efficient and accurate target tracking method than conventional graph-based methods [1, 12]. Image processing techniques map physical node location onto a grid representing the coverage region and then assign nodes an occupation based on their location in the grid. This improves efficiency in part by activating the sensing components of perimeter nodes and placing redundant nodes in an energy preserving sleep state until the perimeter is breached. The Optimized Communication and Organization (OCO) method [1, 12], for example, efficiently secures the perimeter of the detection region and reorganizes the network when a node is damaged or lost. The OCO proposal segments network lifetime into four phases.

Upon mote deployment, the sensor network enters a “Position Collection” phase. The base station broadcasts a message to all nodes and requests that they report their ID and position. The base's neighbors acknowledge the request and broadcast a request for the ID and position of all their neighbors. The process repeats until all nodes are accounted for. The base station maps each node's unique ID and location onto a map representing the surveilled region.

The next phase, “Processing”, selects the minimal set of sensor nodes required to cover the detection region. It achieves this with a three-step process for finding non-redundant nodes. OCO defines a redundant node as one whose sensing radius overlaps with the sensing radius of other nodes in the network. First, the base station initializes an empty list of non-redundant nodes and adds itself to the list. It then analyzes the map to identify sensor nodes with non-overlapping coverage and adds them to the list. The base station then identifies areas of the graph that are not covered by a sensor node from the list. It assigns a nearby node responsibility for that region. All nodes not on the list after this step are considered redundant.

Identification of perimeter nodes (aka border nodes) and construction of the hierarchy tree also occur in the processing phase. The image processor at the base station analyzes each pixel in the coverage map to determine if it is covered by a node from the non-redundant node list. It then assigns pixels in the map that are covered by a node the value 1. Regions of the graph that are not covered by a sensor node are assigned the value 0. When the image processor locates a pixel p with value 1, it checks the value of

neighboring pixels. If any neighbor has a value of 0, then the node encompassing pixel *p* is considered a border node.

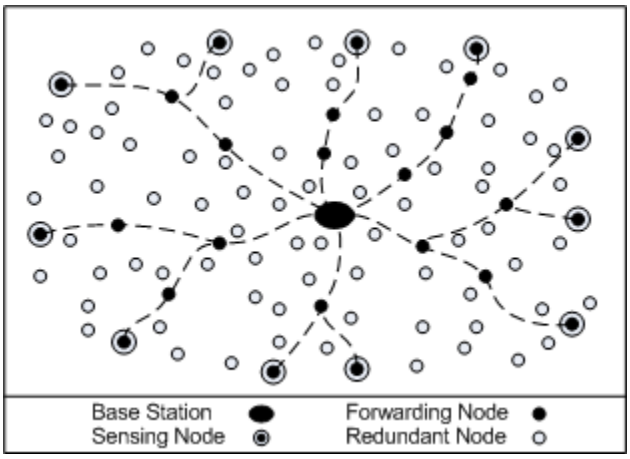

Figure 5: OCO Network Topology

Once border nodes are defined, the base station must find the shortest path from itself to each of the active nodes, including border and intermediate nodes. Border nodes can route alerts to the base station by relaying them through multiple interior forwarding nodes, as shown in Figure 5. After the path is found, the base station broadcasts a message activating the sensor and the radio modules on the border nodes. The base station instructs redundant nodes to enter a sleeping state where their radio and sensor modules are temporarily inactive. Forwarding nodes deactivate their sensor module, yet keep their radio receiver on.

Once the network topology is defined, the network enters the “Target Tracking” phase. Intruders are assumed to enter the surveilled area from outside the perimeter. When a node detects an intruder, it sends a report to the base station via the forwarding nodes. It

will continue to send periodic alerts to the base station while the intruder remains within its sensing radius. The neighbor notification strategy depends on the capabilities of the sensor. The sensing node will also alert its neighbors of the intrusion. Nodes capable of tracking multiple objects will only send two alerts to its neighbors; one when an intruder enters the coverage area and another when the intruder exits. A node that can only track one object at a time will periodically broadcast alerts to its neighbors while an intruder is present. When the neighbors receive an alert, they activate their sensor modules and try to determine if the intruding object has entered their sensing radius.

The “Maintenance” phase of OCO manages reconfiguration of the network when a node is destroyed, moved, or depleted of power. To detect damaged nodes, status messages are periodically exchanged between parents and children. When a node fails to send the status message on time, its peer assumes that it has been damaged. The node that was expecting to receive the status message will notify the base that its peer has vanished. When nodes move, a repositioned node will broadcast an alert stating that it has been moved. Any nodes receiving this alert will forward the message to the base. When a node detects that its power level has fallen below a predefined threshold, it sends a notification toward the base. When the base station receives notification of damage, depleted power, or movement of a node, it will trigger a local reorganization algorithm that reorganizes the affected area in order to cover the gap left by the affected node(s). The reorganization algorithm may activate one or more of the redundant nodes.

OCO was evaluated in a network simulator against DC and LEACH. The simulation was run under various scenarios, respectively with no intruders, one intruder, and multiple intruder objects. As exhibited in the simulation experiments, OCO has several strengths. When no intruders are present, an OCO network will outlast a LEACH network. An OCO network with multiple intruders will reach a constant rate of energy dissipation regardless of the number of intruders. The simulation shows that OCO is nearly as accurate as DC in detecting intruders penetrating the border. OCO surpasses other algorithms in terms of least cost per detected object. OCO messages will be defined in detail in subsequent sections.

4 Security in WSN for Surveillance

Security risks in wireless sensor networks include threats to the confidentiality, integrity, and availability of the system. Security mechanisms used on the Internet are not easily adaptable to sensor networks because of the limited resources of the sensors and the ad-hoc nature of the networks. The adoption of efficient algorithms to mitigate security risks has not kept pace with the rate of miniaturization. This section underscores the challenges of securing sensor network communications and illustrates general attacks against sensor networks. It concludes with a thorough risk assessment of OCO.

4.1 Security Goals

Security assessments of any application focus on the five fundamental tenets of information security: confidentiality, origin integrity, data integrity, non-repudiation, and availability. The definitions used in this subsection are derived from [13] and [14].

Confidentiality means the concealment of information from unauthorized entities. Mechanisms used to achieve confidentiality include access control mechanisms and cryptography. Cryptography scrambles, or encrypts, data to generate ciphertext unintelligible to any unauthorized viewer. The data can be made comprehensible to an authorized viewer who knows the secret key. Semantic security implies a stronger guarantee of confidentiality. Semantic security requires that repeated encryption of a message M would yield unique ciphertext each round. This limits the ability of an

eavesdropper to interpret the plaintext even after observing multiple encryptions of the same message. Use of initialization vectors (IVs) seeded with a counter or a non-repeating nonce provides semantic security.

Origin integrity, also known as authentication, refers to the trustworthiness of the source of data. It means that the receiver of a message can trust that the sender of the message is truthfully who it claims. An intruder should not be able to send a fabricated message and have it treated as a legitimate message from a trusted peer. Data integrity means that the user of the data can trust that the content of the information has not been changed in any way by an unauthorized intruder or improperly modified by an authorized user. Since similar mechanisms provide origin integrity and data integrity, they are commonly grouped under the moniker “integrity”. Integrity overshadows other security goals because of its influence on the reliability of the system and its output. In a robust wireless sensor network, the information contained in a message holds a lower priority than the integrity and authenticity of the message. For example, child nodes commonly send “HELLO” messages to parents to inform them that they are still active. Concealing this message is less important than assuring that it originated from a legitimate node and not an impostor who had perhaps compromised or destroyed that node.

Non-repudiation means that the sender of a message should not be able to deny later that he ever sent that message. In the pre-digital world, one achieved non-repudiation with a simple hand-written signature. In cryptography, it implies that authentication and data integrity can be certified with a high level of assurance and it cannot later be refuted.

Non-repudiation is a critical security service and must be guaranteed in applications that involve financial and business transactions, where accountability of actions is critical to ensure success of the applications. Digital signatures provide non-repudiation.

Availability implies that an authorized user should be able to use the information or resource as required. In a wireless sensor network, the wireless communication link must remain available for the network to sustain operations.

Security literature commonly condenses the five security goals into the C.I.A. triad, signifying confidentiality, integrity, and availability.

4.2 Challenges

The lack of efficient authenticated messaging exposes all layers of the sensor network protocol stack to potential compromise. Without link-layer authentication, an attacker may inject unauthorized packets into the network. This may be used to introduce collisions and force legitimate nodes into an infinite waiting state [15]. Network layer attacks against routing protocols give the attacker the ability to cause routing loops, delay messages, or selectively drop messages [16]. Wireless sensor networks deployed for tracking targets provide valuable application layer notifications about the location of the target. Without authentication, the attacker can perpetrate attacks such as dropping intruder notifications, spoofing intruder notifications to create a diversion, or forcing the entire network into a continual state of reorganization.

In wireless sensor networks, the need for integrity surpasses all other security goals. Data integrity and authentication create a foundation for a highly available and trustworthy

network. While many authentication schemes have been conceived for wireless sensor networks, none of them is a panacea. Algorithms for unicast message authentication, for example, do not meet the requirements for authenticating broadcast messages. Similarly, algorithms that mimic the asymmetry of public key systems by dividing time into slots violate the real-time constraints of intrusion notification systems. Results in [17] show that the best solution tailors the authentication mechanism to the requirements of the application.

4.3 Attacks against Sensor Networks

Wood and Stankovic provide a comprehensive survey of attacks against wireless sensor networks and describe strategies that have been used to reduce their impact [15]. The discussions of attacks against sensor networks in the rest of this subsection are mainly derived from their work. The analysis classifies attacks following the OSI reference model for network protocol design.

Physical tampering poses a threat to sensors. If sensors are distributed in an unprotected area, an attacker could destroy the nodes or collect the sensors, analyze the electronics, and steal cryptographic keys. This complicates the process of bootstrapping newly deployed sensors with cryptographic keying material. To protect against this, sensors must be tamper-proof or they must erase all permanent and temporary storage when compromised. Secure key rotation mechanisms can also mitigate the threat of stolen cryptographic keys.

Jamming attacks against wireless radio frequencies affect the availability of the network. While it is most efficient to program sensors to communicate on one specific wireless frequency, an attacker could easily broadcast a more powerful signal on the same frequency and introduce interference into the communications channel. Spread spectrum technologies such as frequency-hopping spread spectrum alleviate the impact of jamming; however, complex channel hopping patterns reduce battery life. Nodes could also try to detect jamming and sleep until the jamming stops, resulting in a temporary, self-induced denial of service (DoS).

Link layer protocols face similarly challenging threats. Attackers can introduce collisions that force communicating nodes to retransmit frames. Following a collision, a node must back-off and wait for the channel to clear before attempting to resend. The attacker can continually introduce collisions until the victim runs out of power. While error-detecting mechanisms suffice for common transmission errors, they do not reduce the influence of maliciously generated collisions. Collisions maliciously injected near the end of a legitimate frame rapidly exhaust the resources of the legitimate node. Authentication cannot alleviate these physical and link layer attacks.

Network layer attacks take advantage of the ad-hoc organization of wireless sensor networks. Any node in the network can become a router, forwarding traffic from one node to another. For example, OCO promotes simple nodes into the forwarding node occupation because of their location on the coverage map. By manipulating routing information, the attacker can shape the flow of traffic. The simplest attack compromises a routing node and

forces it to drop messages, creating a network “black hole”. The attacker can also selectively delay messages routed by the compromised node. In a wormhole attack, the adversary tunnels messages destined for one part of the network through a path under enemy control. Wormhole attacks facilitates eavesdropping, message replay, or disconnection of a segment of the network.

One technique to create black holes circumvents the way routing protocols organize the network. Nodes typically adopt the router that broadcasts route advertisements with the strongest radio signal. This strategy reduces the energy required for a node to communicate with its default router. An attacker can manipulate this strategy to convince legitimate nodes that it requires the least communication overhead. One way to accomplish this is with HELLO floods [16]. To perpetrate a HELLO flood, the attacker repeatedly broadcasts HELLO messages at higher power than every other node. Hello floods can be used by an attacker to persuade the network nodes that it is a legitimate neighbor and a reliable next hop. An attacker can also corrupt shared routing tables by spoofing, manipulating, or replaying routing messages.

Internet style attacks have their analogue in wireless sensor networks. Misdirection attacks, such as the Internet smurf attack, work in sensor networks. The attacker can send multiple messages to broadcast addresses with a source address forged to the intended victim's address. The broadcast responses will overwhelm the victim, flood its communication channel, and drain its power. Filtering the legitimate messages from the responses in a smurf attack requires a hierarchy not present in many wireless sensor

network routing protocols. A similar attack, called a Sybil attack, targets systems that select peers based on their reputation. In a Sybil attack, the adversary sends a large number of fabricated messages that appear to be forwarded from other nodes. Legitimate nodes begin to trust the attacker because it seems to fairly route traffic. The legitimate nodes will eventually adopt the adversarial node as their router.

Transport-layer protocols provide end-to-end connectivity between nodes. Sequencing, such as that used in the Transmission Control Protocol (TCP), improves the reliability of the connection. Protocols that use sequencing may succumb to Denial of Service (DoS) attacks. The classic TCP SYN flood applies to sensor networks. An adversary can flood the victim with synchronization requests and limit the ability for other nodes to communicate with the victim. One solution limits the number of synchronization requests accepted, but this limits both adversaries and allies. Client puzzles, a more complex solution, require the client to make a commitment to the server before it is allowed to initiate a conversation. When the client initiates a connection, the server will respond with a puzzle that the client must solve. The client must complete the puzzle and send the answer to the server before the server will accept a full connection. While this solution protects the server from SYN floods, it may harm allies that have fewer computational resources than the adversary does.

Origin authentication and message integrity can mitigate attacks at the network layer and above. Threats such as spoofing or fabrication of routing information justify the need for origin and data integrity of even the simplest HELLO messages.

4.4 OCO Risk Assessment

OCO provides both network and application layer functionality, making it susceptible to many of the attacks previously described. As a first step in designing an appropriate security solution, this subsection formally evaluates the security threats to confidentiality, integrity, and availability against OCO. It identifies the purpose , path, and total risk of the 14 OCO messages as defined in [12].

The complexity of a risk assessment lies in defining the asset one is trying to protect and identifying attack vectors against that asset. In a conventional model, the assessment may focus on intellectual property or personally identifiable information as it is stored, processed, or transmitted. In a wireless sensor network, one can abstract the definition of the asset from the goals of the application. OCO aims to provide perimeter intrusion detection, target tracking, and network longevity. The OCO messaging architecture summarized in Table 1 supports the ability for OCO to achieve its target tracking and energy efficiency goals.

Table 1: OCO Message Summary

Message	Purpose
M1	The sender seeks child nodes for adoption
M2	Nodes acknowledge their adoption
M3	The base informs child nodes of the id of their parent
M4	The base informs parent nodes of the id of their descendant
M5	The base informs border nodes of their occupation
M6	The base assigns redundant nodes to a sleep state
M7	The base informs routing nodes of their occupation
M8	A sensing node alerts the base station of an intrusion
M9	A sensing node alerts its parent of an intrusion
M10	A child node reports its health to its parent
M11	A parent advertises its health to descendants
M12	A child node reports that is has lost its parent
M13	A parent node reports that is has lost its child
M14	The base resynchronizes redundant nodes

A risk assessment enables development of security countermeasures tailored to protect the assets most exposed to risk. Risk assessments should also evaluate the cost of the countermeasures to determine whether the countermeasures cost more than the value of the asset. The simulation in Section 8 and Section 9 quantitatively evaluates the cost of securing OCO. If the cost of the countermeasures exceeds the value of the asset, the system owner may choose to accept the risk and leave the system in its original state. Any security countermeasure designed for OCO must protect the network's most important asset, the ability of the base station to receive alerts when an intruder penetrates the perimeter. The security mechanism must secondarily protect OCO's most significant achievement, preservation of the energy resource. This assessment ranks the risk of OCO from the perspective of information transmitted through its messages. Table 2 summarizes the results of the OCO risk assessment. The risk assessment formula used in this analysis is [18]:

Risk = Value of the Asset x Severity of the Vulnerability x Likelihood of an Attack.

Table 2: OCO Risk Matrix

Message	Asset Value	Vulnerability Severity	Likelihood of Attack	Risk
M1	10	10	2	200
M2	10	7	1	70
M3	10	7	10	700
M4	7	4	10	280
M5	8	9	10	720
M6	8	9	10	720
M7	8	9	10	720
M8	10	5	10	500
M9	9	5	10	450
M10	7	9	4	252
M11	7	9	4	252
M12	7	5	10	350
M13	7	4	10	280
M14	7	4	10	280

The assessment assigns each risk factor a weight of 1 through 10, with 10 being the highest. Multiplication of the factors yields an aggregate security risk value. The assessment in subsections 4.4.1 through 4.4.4 demonstrates why attacks against integrity rank higher than attacks against confidentiality.

4.4.1 Position Collection Phase

OCO possesses its own routing protocol optimized for target tracking. Messages transmitted throughout the Position Collection and Processing phases define the routing topology and assign nodes an occupation. Attacks against these messages may allow the attacker to insert themselves as a Man-in-the-Middle (MITM) between legitimate nodes and the base station. Messages *M1* and *M2* belong to the Position Collection phase, which collects node location and id in order to populate the coverage map. This phase only occurs upon node deployment.

Network organization starts when the base station broadcasts message *M1*. This message announces that the sender seeks child nodes for adoption. Orphan nodes that receive this message will accept the broadcaster as their parent, flag themselves as adopted, and broadcast the message to their neighbors. Message *M1* is recursively rebroadcast throughout the network until all nodes have been adopted. Successful forgery of *M1* with precise timing may allow the attacker to become a MITM between all legitimate nodes and the base station. This enables the attacker wholly control network operations and notifications. However, the attack must be timed to occur between node deployment and the first broadcast of message *M1* by the legitimate base station. Once nodes have been adopted, message *M1* is no longer accepted. While *asset value* and *vulnerability severity*

both deserve a high rating of 10, the complexity of timing this attack limits its likelihood to a rating of 2. *M1* receives a risk rating of 200, as summarized in Table 2.

Nodes acknowledge their receipt of *M1* and their adoption by sending message *M2* to their new parent. The message includes child node id and node position. The parent records the id of its child and forwards the message toward the base station. The base station utilizes the node id and position during the Processing phase to draw a map of the network. Note that the parent assigned in the Position Collection process may not be the same parent assigned during the Processing phase. The parent assigned during Position Collection simply provides a path so that nodes can report their position to the base. If an attacker spoofs *M2*, the rogue node's id and position will be reported to the base along with all the legitimate nodes. This gives the attacker a limited chance of being accepted into the network as a legitimate node and perhaps even selected as a forwarding node. The attacker has equal chances of being selected as border node, or even a redundant node. An attacker could increase their chances by deploying multiple adversarial nodes into the legitimate node pool. If the attacker could insert itself in the middle of the transaction, it may also modify the node position field in legitimate messages. This attack could damage the integrity of the map and cause the base station to produce a fragmented, inefficient design. Attacks against *M2* would not allow an adversary to play MITM between the base station and legitimate nodes as with attacks against *M1*. This vulnerability must be exploited during the Position Collection phase in order for the attacker to benefit. While the *asset value* remains 10, the vulnerability, which may only affect a subset of perimeter nodes, receives a severity rating of 7. The attack shares the timing constraints with M1 and the

results can be unpredictable. *Likelihood of attack* receives a 1. *M2* receives a risk rating of 70.

4.4.2 Processing Phase

The base station sends messages *M3* through *M7* once it has completed the image processing analysis on the coverage map. During Processing, the base station analyzes node distribution and determines optimal organization of forwarding nodes, sleeping nodes, and border nodes. Successful and unaltered delivery of these messages assures complete coverage and minimal redundancy. In the current design of OCO, once a node receives their assignment in the Processing phase, they may continue to accept new assignments from the base station because of maintenance. This increases the attack window and significantly increases the opportunity for an attack.

M3 is broadcast by the base to notify each descendant of its parent. This defines the path from the border to the base, assuring that border and forwarding nodes can send alerts and status reports back to the base station. Attacks against this message can take a significant toll on the accuracy of the network. An attacker can forge or manipulate *M3* and replace the parent id field with its own id or the id of one of its adversarial counterparts. When child nodes accept an adversary as a parent, the child will send later alerts and health messages to the adversary, who may act as a MITM. The adversay can drop or ignore these intrusion notifications. Since this attack influences delivery of alerts and allows false negatives, *asset value* receives a rating of 10. Since it is limited in scope to targeted nodes and not all legitimate nodes, vulnerability is set to 7. The *likelihood of attack* is set to 10 since the attack window spans almost the entire network lifetime.

M4 is broadcast by the base to notify each forwarding node of its descendants. This message defines the path from the base down the tree to the border nodes. The route down the tree comes into play during the Tracking phase. When a border node detects an intrusion, it alerts its parent, a forwarding node. The forwarding node in turn alerts its other immediate descendents. Both the forwarding node and its descendants activate their sensor to track the intruder. If forwarding nodes are provisioned with a modified set of children, the network may loose track of the intruder as it moves inside the coverage area. Attacks against *M4* only affect the ability of a network to track an intruder, not its ability to alert the base station. The original perimeter node can still detect an intrusion and alert the base, thus, *asset value* receives a rating of 7. The vulnerability severity receives a 4. Because of the broad attack window, the *likelihood of attack* remains at 10.

Messages *M5*, *M6*, and *M7* each serve a similar purpose, notifying each node of its occupation as a forwarding node, a sleeping node, or a border node. Forgery or modification of M5, M6, and M7 may allow an adversary to maliciously create a network with no border nodes, thus disabling the perimeter. The adversary could put all nodes to sleep by assigning them a redundant node occupation. The attacker could take the opposite approach and turn all nodes into border nodes with active sensors, thus creating a LEACH network that would quickly loose all power. The ability to modify node occupation makes this a serious threat to the network's ability to track intruders and on its efficiency. The *asset value* receives a rating of 8. This critical vulnerability is rated at 9. The *likelihood of attack* remains at 10 as with other messages in the Processing phase.

4.4.3 Tracking Phase

Messages *M8* and *M9* constitute the Tracking phase. When a border node detects an intrusion, it must send two alerts. It will send *M8* with the ultimate destination of the base station and it will broadcast M9 to its neighbors. *M9* will only be accepted by forwarding nodes, which will then activate their sensor component. Modification or forgery of these messages results in false positives, which damage the trustworthiness of the network. These false positive errors reduce the effectiveness of the network and create diversions. For example, imagine a sensor network monitoring the perimeter of a battlefield. An intruder may send a spoofed *M8* to the base station. When the base receives the alert of an intruder crossing the perimeter, the commander immediately sends reinforcements to the location to defend the intrusion. An impostor border node could send an alert on one side of the battlefield as a diversion while real intruders cross the perimeter from the other side. Thus, the severity of a false positive depends on the process for responding to alerts. The asset attacked remains the intrusion detection capabilities. Spoofing or fabrication of *M8*: Asset value 10, Vulnerability Severity: 5, *Likelihood of attack*: 10. Spoofing or fabrication of *M9*: Asset value 9, Vulnerability Severity: 5, *Likelihood of attack*: 10.

4.4.4 Maintenance Phase

The Maintenance phase promotes reliable network operations by providing node health messages, topology change alerts, and synchronization updates for sleeping nodes. This phase assures that damage to or movement of nodes does not significantly influence the ability of the system to detect and track intrusions. When changes to the topology occur, the Maintenance system will call upon the Processing algorithm to review the map and find redundant nodes to fill the role of changed border or forwarding nodes.

With message *M10,* a child node reports its energy level to its parent. When the energy level falls below a threshold, the parent will alert the base station to initiate Maintenance. Parent nodes use *M11* to broadcast to their children that they are still active. A sophisticated attacker could replace the energy level in reports from the perimeter nodes with a higher value. When the perimeter node finally dies, the attacker could spoof messages from the dead node and keep the network from initiating a border repair algorithm. Since such attacks against M10 would prevent alerts from being sent, it receives a 9 in asset value. The severity of the vulnerability receives a 7. The attack requires a sustained effort and thus receives a likelihood of 4. A similar attack could be used against *M11*. An attacker that compromises or destroys a forwarding node must continue to send spoofed *M11* status reports in order to keep the descendant nodes for broadcasting an alert that they lost their parent. The adversary could then drop alerts from its descendents. Message *M11* receives the same risk rating as *M10*.

Messages *M12* and *M13* contain S.O.S messages used to initiate a border repair phase. When a node has lost its parent, it broadcasts this S.O.S. message to initiate the adoption process. These messages initiate a local border repair process, where the base station re-runs the processing algorithm to find a substitute for the damaged nodes. An adversary can spoof messages, making them appear to originate from legitimate node. For example, an attacker could fake *M12* messages from parent nodes stating that they have lost their children. The base station will respond by flagging those nodes as dead and find nearby redundant nodes to replace their functionality. In a densely populated OCO network, this will slowly erode the perimeter. Fabrication of both *M12* and *M13* will

deplete power resources. The *asset value* of message *M12* is ranked at 7. The severity of this vulnerability receives a 5. The likelihood of this attack occurring is 10, giving message *M12* a total risk rating of 350. Message *M13* has a similar *asset value* and *likelihood of attack*, 7 and 10. The severity of the vulnerability, however, is only a 4, yielding a total risk rating for message *M13* of 280.

The base station uses *M14* to send control messages to redundant nodes. These include updates on the "time to synchronize" and "time to stay awake" parameters. Spoofing or fabrication of *M14* does not take an immediate toll on the system; however, it does provide a mechanism to eliminate the benefits of redundant nodes. An attacker can force sleeping nodes to permanently sleep, thus eliminating the healing abilities in OCO, or it can force the sleeping nodes to permanently stay awake, which would cause them to keep their radio transmitter on wasting power. The *asset value* provided by message *M14* is ranked at 7. The severity of this vulnerability receives a 4. Since the attack window spans most of the network lifetime, the likelihood of this attack occurring is 10. The total risk rating of message *M14* is 280.

4.4.5 Attacks against Confidentiality

Attacks against system confidentiality rank low in the OCO risk assessment. OCO messages do not contain sensitive information that could expose the network to harm if it were exposed. The messages simply contain source and destination addresses, node position and energy, and synchronization schedules. An attacker can use other methods besides attacking OCO to discover this data. Thus, threats against OCO message confidentiality rank low and are excluded from this assessment.

5 Survey of Message Authentication Protocols

This section summarizes some of the most relevant proposals that integrate origin integrity and data integrity in to wireless sensor network communications. Each proposal possesses unique qualities that influence its applicability to the OCO target tracking method. Many combine techniques for origin integrity and message integrity with other security goals, such as confidentiality or replay protection. However, these features may consume excessive processor, storage, or energy resources. This section concludes by contrasting the cost of each mechanism.

A number of requirements frame the analysis of authentication protocols for OCO. First, an authentication protocol should be resistant to node compromise by allowing secure key management. The protocol may provide an integrated key-rotation mechanism or allow for key rotation by an external module.

In addition, the protocol must have low computation overhead for both the sender and the recipient of a message. This analysis measures computation overhead in terms of the number of processor instructions, the amount of code memory, and the amount of data memory of the major authentication protocols. The protocol must also require low communication overhead. On some mote platforms, the radio transceiver consumes more energy than the processor.

Finally, messages supporting the authentication protocol must function in an unreliable network. In target tracking networks, alarms are time-sensitive. Thus, the protocol should support the ability to immediately authenticate a message upon receipt.

5.1 Cryptographic Constructs

The authentication proposals presented later in this section differentiate themselves on factors including key management, packet format, and selection of block ciphers and modes of operation. This subsection presents the cryptographic foundation for the comparison of the proposals.

5.1.1 Conventional Authentication

The roots of message integrity begin with cryptographic checksums, also known as hashes. These checksum functions take a message and condense it into a smaller message digest [13]. The simplest example, the parity bit, counts the number of 1-bits in a message to produce a checksum of 1-bit in length. Strong cryptographic hash functions must possess three desirable properties. First, the hash must be easy to compute, not consuming significant computational resources. Second, it should be computationally infeasible to reverse the hash function. This means that given the result of the hash $h(M)$, one should not be able to determine M. A third desirable property of hashing algorithms states that two distinct messages, when hashed, will yield two distinct checksums. However, according to the pigeonhole principle, there is a chance that two distinct messages M and M', will yield produce the same hash value, $h(M) = h(M')$. This condition, known as a collision, can be exploited to defeat hash functions [19]. The MD5 [20] and SHA-1 [21] hash functions are employed in several security applications and protocols. MD5 condenses a message into a

hash of 16 bytes. SHA-1 condenses a message into a 20-byte hash. Both MD5 and SHA-1 have been proven susceptible to collisions [19, 22].

Hash functions provide a level of message integrity between communicating peers. A sender prepares a message *M* and calculates the checksum $x = h(M)$. It then sends the checksum along with the message to the recipient. When the recipient receives message *M,* he can recalculate the checksum on the received message *M*. If the checksum appended to the message matches the checksum calculated by the recipient, then the recipient can be assured of message integrity. Figure 6 illustrates this process.

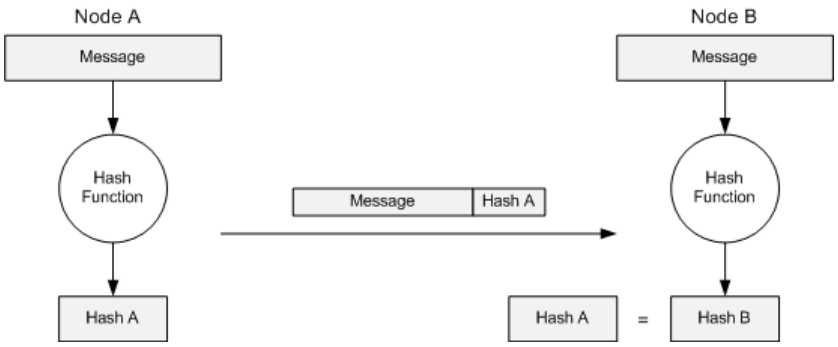

Figure 6: Hash Function

Cryptographic checksums cannot provide assurance that messages arrive without modification or that they originate from an authentic sender. Since an attacker may know the hashing algorithm in use, an attacker could simply replace message *M* with message *M'*, compute the hash $x' = h(M')$, and send the concatenation of the message *M'* and the hash *x'*. The recipient will calculate the hash of *M'*, which will match the *x'* sent by the attacker. Thus, the recipient cannot validate that authenticity of the message. Message authentication codes (MAC), an instantiation of hashes that uses a unique key, provide both

the data integrity of checksums and origin integrity provided by a secret key. Both the sender and receiver should share the key. If an adversary learns the secret key, the hashing function is compromised.

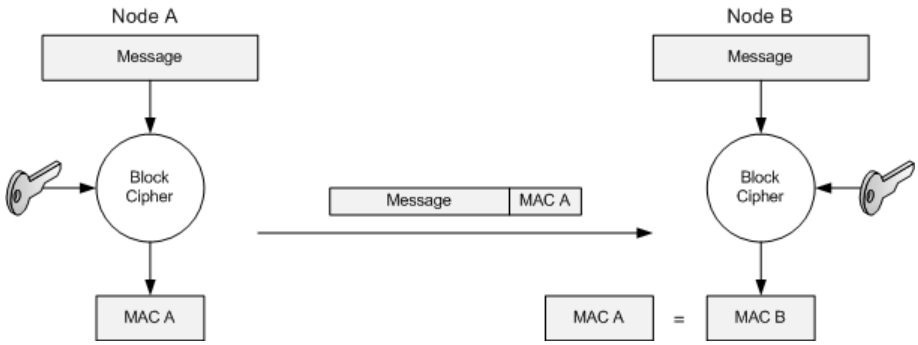

Figure 7: Message Authentication Code

As illustrated in Figure 7, a MAC is constructed by encrypting a message with a block cipher in Cipher Block Chaining (CBC) or Cipher Feedback Modes (CFB) [14]. Use of the Cipher Block Chaining mode to generate a MAC is commonly known as CBC-MAC. Many WSN authentication mechanisms employ CBC-MAC. However, the CBC-MAC operation has been shown to be insecure for variable length messages [23]. The U.S. NIST has approved an implementation of CBC-MAC that securely authenticates variable length messages. This mode, known as CMAC [24, 25], has not yet been reviewed in wireless sensor network research.

5.1.2 Unicast vs. Broadcast Authentication

Unicast authentication provides the assurance of origin integrity when a message is sent from one sender to one receiver. A message authentication code (MAC), generated by the sender/creator of the message by using a secret key, can be used to ensure origin

integrity. For unicast messages, static symmetric (shared) key cryptography meets the requirements because the two peers are trusted not to leak the key. The speed and efficiency of symmetric key cryptography suit the constraints of wireless motes.

Broadcast authentication ensures that multiple recipients of a message can verify its origin integrity. If using MACs to ensure broadcast authentication, all recipients of the message must share the symmetric key. The unique challenge for broadcast authentication involves the management of that shared key. If the key is broadcast to potential recipients, an adversary could eavesdrop on the key broadcast, capture the key, and generate a legitimate MAC for a forged message. Public key cryptography solves the problem of securely sharing a key for conventional Internet computing systems. However, public key cryptosystems consume far too much storage, computation, and bandwidth resources to be applied in wireless sensor networks. The research surveyed throughout this section introduces mechanisms that either replicate the asymmetry of public key cryptography or provide efficient key rotation.

5.1.3 Block Ciphers

Symmetric key cryptography consists of two categories of ciphers: block ciphers and stream ciphers. Stream ciphers operate on a single bit or byte at a time. Block ciphers operate on groups of bits called blocks [14]. Common block ciphers considered for wireless sensor networks accept block sizes of 32, 64, and 128 bits. Authentication mechanisms typically employ block ciphers because they can be used to generate MAC. Three block ciphers stand out when analyzing cryptographic ciphers for sensor nodes: RC5 [26], Skipjack [27], and AES [28].

The cost of these ciphers on wireless sensor nodes has been scrutinized in [29-32]. Roman, et al., evaluated the purported benefits of implementing block ciphers in hardware. They reveal that software implementations suffice because the greatest cost arises from communication overhead, not processing or storage. In Roman's studies, the Skipjack cipher excels at providing efficient block cipher operations because of its speed and simple key schedule. Roman shows that Skipjack has an average encryption time of 48 microseconds per byte. Law further improved its implementation to reach 25 microseconds per byte. Despite its efficiency, the small 80-bit Skipjack key limits the algorithm's resistance to attack. The stronger AES algorithm, based on the Rijndael cipher [33], accepts key sizes up to 256-bits. Law and Choi separately show that AES has an average encryption time of 50 microseconds per byte. This additional processing overhead can bottleneck high bandwidth communications. However, AES may be appropriate for target-tracking applications since they do not require high bandwidth. Großschädl, et al., show that existing implementations of AES can be easily optimized to further reduce energy and storage consumption. Table 3 summarizes the block and key sizes of common block ciphers.

Table 3: Common Block Ciphers

Cipher	Key Size (b)	Block Size (b)
AES	128/192/256	128
RC5	0 ~ 2040	32/64/128
RC6	128/192/256	128
Twofish	128/192/256	128
Skipjack	80	64
XTEA	128	64

5.2 SPINS

In [3], Perrig, et al., propose a suite of security protocols called SPINS (Security Protocols for Sensor Networks) that provides message authentication, data integrity, confidentiality, and replay protection. The SPINS suite includes two protocols: SNEP and µTESLA. SNEP provides unicast authentication, confidentiality, and replay protection. µTESLA offers a solution for authenticated broadcast messaging in sensor networks. These two protocols specifically address self-organizing wireless sensor networks, like OCO, that have a multi-hop routing topology. The proposal secures three communication paths: single node to the base station, base station to a single node, and base station broadcast to all nodes. SPINS does not offer a solution for node-originated broadcast messages.

5.2.1 SNEP

The SNEP component of SPINS packages data authentication, protection against replay, and weak data freshness into one cryptographic protocol. SNEP provides origin authentication and message integrity by appending an 8-byte MAC to the ciphertext. It generates the MAC by running the plaintext message and the counter through the RC5 block cipher in CBC-MAC mode. Since the MAC provides message integrity, the CRC field is dropped. Thus, total packet length increases only by 6 bytes.

SNEP requires communicating endpoints to maintain a shared counter as a mechanism to prevent replay attacks. Sensor network protocols commonly avoid counters because they increase communication overhead. SNEP initializes the counter when two nodes begin communicating and increments it after each communication block, thus eliminating the need to send it along with each message. While this minimizes

communication overhead, it requires the nodes to set aside additional memory for the counter.

The counter aids in both in the encryption process and in calculation of the MAC. Use of the counter in the encryption process provides semantic security, which means that repeated encryption of a message *M* would yield unique ciphertext each round. Semantic security limits the ability of an eavesdropper to interpret the plaintext even after observing multiple encryptions of the same message. Use of the counter in calculation of the MAC provides protection against replay attacks. The counter also helps enforce weak message freshness. Message freshness describes the level of assurance that a message to a node *A* was created by node *B* in response to a request from node *A*. The authors employ a nonce in cases where strong message freshness is required. Node *A* generates the nonce and sends it in the request to node *B*. Node *B* implicitly returns the nonce to node *A* by encoding it into the MAC.

5.2.2 µTESLA

While SNEP protects unicast messaging, secure broadcasting messages in a wireless sensor network remains a complex problem. In Internet communications, asymmetric or public key cryptography serves as the foundation for broadcast authentication. Public key cryptography requires too many computation cycles and too much storage for sensor nodes. Symmetric key cryptography requires strong protection for the shared key. Once the shared key is disclosed, an impostor could use it to spoof messages from the legitimate sender. As part of SPINS, µTESLA manages this problem by using a chain of symmetric keys that are periodically rotated. This mechanism works best

with broadcast messages sent by a base station. For a node to send broadcast messages, an intermediary must intervene.

To accomplish base station originated broadcast authentication, the base station first randomly selects a key, K_n. It then successively applies a one-way hash function F to K_n, generating keys K_{n-1} through K_0. The base station rotates keys on a design specific interval. When the base station broadcasts a message, it appends a MAC to the message that was generated with key K_i. Existing nodes that receive the message can immediately authenticate it by verifying the MAC with key $K_i = F(K_{i+1})$. Any recipients that are new to the network cannot authenticate the message until the next interval, when the base station broadcasts the new key, K_{i+1}. The recipient will calculate K_i by applying the one-way hash function to K_{i+1}. Note that this use of key rotation based on time intervals requires the base station and the nodes to be loosely time synchronized. If nodes loose synchronization with the base station, validation of the MAC will fail.

In SPINS, wireless sensor nodes cannot send authenticated broadcast messages without the assistance of the base station. The authors propose two strategies. First, the node can send the message to the base station and allow the base station to broadcast it. Second, the node can broadcast the message and let the base station manage distribution of keys.

The implementation of both SNEP and µTESLA calls upon the RC5 cipher. For message authentication and generation of random numbers, the authors employ CBC-MAC. Encryption and decryption occur with RC5 in counter (CTR) mode, a mode of

operation that turns a block cipher into a stream cipher. Since the same function provides encryption and decryption, the algorithm preserves code memory. Other advantages of RC5 include its efficiency, its avoidance of multiplication, and its small lookup tables. Rijndael and other block ciphers require complex calculations and memory to store large lookup tables.

SPINS addresses the security requirements for a sensor networks without real-time constraints. This system provides confidentiality for unicast messages and message authentication for both unicast and broadcast messages. The system minimizes utilization of energy, storage, and bandwidth resources. However, the SPINS model does not efficiently handle node-originated broadcast authentication, nor does not scale to all sensor network topologies. Most importantly, SPINS does not provide a solution for instantaneous authentication, which is required for alerts such as those in OCO.

5.3 RPT and LEA

μTESLA requires the recipient to wait for an interval of time before it can determine the key to authenticate a previously received message. This limits its usefulness in sensor networks with real-time constraints such as target tracking networks. Intrusion notifications may occur irregularly, but need to be authenticated immediately. When keys are sent more frequently than they are used, the recipients expend computation cycles working through the key chain. Luk, et al., introduce in [17] RPT (Regular and Predictable Times) and LEA (Low Entropy Authentication) as solutions to these issues.

They begin by suggesting three modifications to µTESLA that make it more suitable for infrequent, but urgent alarms. The simplest modification reduces the key disclosure interval. However, this approach wastes energy since the recipients must process the more frequent key distribution messages. Another solution proposes publication of multiple keys in a single message. A third solution replaces the one-way hash chain with a hash tree. A tree with N keys will grow to a height of $d = log_2 (N)$. Only d values must be sent with each message to verify that it is authentic. Hash chains can be combined with hash trees, providing a link from one chain the next. The leaf node of one tree can be used to compute the hash chain appended to the subsequent branch. There are two advantages to this strategy. First, messages can be authenticated using the one-way hash of the last known key as with standard µTESLA. Second, if no messages were sent in a long period, the recipient can use the hash tree leaves as a shortcut through the hash chain.

The RPT protocol, proposed by Luk, et al., efficiently authenticates messages sent at Regular and Predictable Times [17]. RPT breaks time into short intervals and assigns a key from a one-way key chain to each interval. The sender calculates δ, the sum of the maximum propagation delay and the maximum time synchronization error. To send a message, the sender first broadcasts only the MAC of the message. The sender then waits δ, and sends the message and the key used to generate the MAC. The receiver verifies that the key is still fresh, and then verifies that the MAC of the message matches the MAC originally broadcast by the sender. This allows for sparse key rotation and reduces communication overhead caused by key distribution. This approach benefits circumstances where the message contents are known well in advance, and some procedure requires that

the message be sent regularly and on schedule. For example, consider time synchronization signals. A base station may send a synchronization and key rotation message every day at noon. With standard μTESLA, the message could not be authenticated by a new member of the sensor network until a key rotation the following day.

The second protocol introduced by Luk, et al., Low Entropy Authentication (LEA) [17], provides security for short message that change infrequently. LEA evolved from one-time signatures like the Merkle-Winternitz construction. One-time signatures include private and public keying information connected by a one-way hash function. The function starts with a pseudo-random private key at the root of a directed acyclic graph. Two edges connect this private key to two vertices: the one-way hash of the key and a checksum of the key. These two vertices are again repeatedly hashed so that the length of the chain is sufficient to sign the message. A message of x bits requires a chain of $2x$ values long. At the end of the chain, the hash of the key and the hash of the checksum are concatenated to form the public key.

As with other asymmetric key algorithms, the sender of a message signs the message with their private key. The length of the signature depends upon the length of the message to be signed, thus one-time signatures such as Merkle-Winternitz suit short messages with low entropy. While this protocol efficiently generates and verifies signatures, it does not scale well for signatures of long messages. Additionally, one-time signatures require a unique, authentic public key for each message. This challenge can be overcome, however, by distributing the public keys far in advance of the signed message.

The sender could potentially send a set of *n* keys to sign the next *n* messages. Perrig and Luk suggest using their own RPT protocol for key pre-distribution. This approach suffers since it requires the recipient to store *n* keys until they are actually used.

5.4 TinySec

TinySec [8] aims to satisfy three security goals: origin integrity, message integrity, and message confidentiality. It achieves these goals while limiting the impact on computation, memory usage, and bandwidth. The TinySec approach to securing wireless sensor networks provides message integrity and confidentiality in a way that facilitates integration into sensor network applications. TinySec provides two modes of operation: TinySec with origin and message authentication only (TinySec-Auth) and TinySec with authentication and encryption (TinySec-AE). TinySec, the first proposal to make symmetric key encryption primitives available to sensor nodes [29], now comes bundled in the TinyOS operating system for wireless sensor nodes. This eases integration into application development and increases the likelihood of deployment of secure wireless sensor networks.

TinySec makes use of message authentication codes (MACs) to provide authentication and message integrity. The sender and receiver share a secret key to cryptographically sign messages before their delivery and to validate messages upon receipt. The recipient validates the signature to detect tampering or damage incurred during transit. TinySec authenticates a packet with a 4-byte MAC. The MAC authenticates the destination address, AM type, length, and payload TinyOS fields. This MAC replaces the CRC field at the end of a TinyOS packet since a MAC can detect both malicious changes

and transmission errors. Use of a shared authentication key allows the TinyOS Group ID field to be dropped. Since TinySec drops these fields, TinySec-Auth only results in 1-byte of additional communication overhead.

Use of a 4-byte MAC illustrates the tradeoff between risk and the cost of a security countermeasure. MACs usually range from 8-bytes, as in SPINS, to 16-bytes in length. The TinySec authors claim that a 4-byte MAC meets the requirements for wireless sensor networks. This length gives adversaries a 1 in 2^{32} chance of successfully brute forcing a MAC for a particular message. To succeed in this forgery, the adversary must send messages to the target recipient. The limited wireless data transfer rate on wireless sensor networks only allows for 40 forgery attempts per second. It would take over 20 months to send all 2^{32} possible MAC combinations. The recipient sensor would likely run out of power before this attack could complete. As a compensating control, nodes should alert the base station when the rate of MAC failures exceeds a predefined threshold.

The TinySec MAC evolved from the cipher-block-chaining MAC (CBC-MAC) block cipher mode of operation. Bellare, Killian, and Rogaway have demonstrated that this construction fails to secure variably sized messages [23]. They suggest three alternatives for generating MACs of variably sized messages. The variant used by TinySec XORs the encryption of the message length with the first plaintext block. The authors considered the RC5 and Skipjack block ciphers the most appropriate cipher for embedded microcontrollers. They implement TinySec with Skipjack because RC5 requires additional RAM to store a pre-computed key schedule.

TinySec-AE offers both basic confidentiality and semantic security, preventing adversaries from gaining partial knowledge of the plaintext by observing repeated encryption of the same message. Semantic security requires use of initialization vectors (IVs), which increase diversity in the plaintext. Target tracking networks like LEACH, GDAT, and OCO, commonly exhibit low message entropy. Wireless sensor network that require confidentiality often include mechanisms to provide semantic security. TinySec uses the CBC mode of operation with the Skipjack cipher to provide confidentiality. Unlike SPINS, TinySec views keying mechanisms and cryptographic functions modularly. This enables use of a variety of keying mechanisms within TinySec, including a simple network-wide shared key.

Security mechanisms must be easy to integrate into an application; otherwise, they will not see widespread deployment. The security, performance, and usability of TinySec can be enabled in sensor network applications by setting a compile-time flag in a TinyOS makefile. Developers can select TinySec-Auth or TinySec-AE and tune the level of security around their application's requirements. TinySec simplifies integration with a design focused on transparency and portability. TinySec achieves transparency by assuring that there is consistency between standard network APIs and network APIs that enable security. This facilitates upgrades of legacy applications to support security countermeasures. Implementation of TinySec as a link layer module makes this transparency possible. The TinyOS radio stack sends all radio events to the TinySec module. In support of portability, TinySec aims to run on a variety of platforms that support TinyOS.

5.5 MiniSec

The research team that introduced SPINS and µTesla propose a wireless sensor network security architecture called MiniSec [34] that improves efficiency in delivering authentication and confidentiality to wireless sensor networks. MiniSec secures both unicast and broadcast communication. MiniSec's most significant contribution rests in its use of the same block cipher operation to provide confidentiality and authentication. MiniSec also introduces a novel semantic security strategy that only requires transmission of a fragment of the initialization vector (IV). MiniSec capitalizes on advancements in mote hardware, leveraging increasingly available mote memory to defend against replay attacks.

MiniSec proposes significant energy savings for wireless sensor networks that require both confidentiality and integrity. TinySec minimizes memory and energy use by omitting replay protection and using a single network shared key. ZigBee takes the opposite approach and maintains a high level of security by sending an 8-byte IV [35]. The authors claim that MiniSec consumes 1/3 the memory of TinySec-AE, the authenticated encryption mode of TinySec.

MiniSec employs a block cipher mode known as Offset Code Book (OCB), a block cipher mode of operation developed by Phillip Rogaway [36]. OCB provides authentication and encryption in one pass. In comparison, TinySec and ZigBee require two block cipher passes: one for confidentiality and another for authentication. OCB takes in the message, the key, and a non-repeating nonce, and concurrently generates the ciphertext and a tag used for authentication. The nonce provides semantic security, assuring that any two identical plaintext messages encrypted with the same key yield different ciphertext. The tag

functions as a message authentication code (MAC). Unlike other block cipher modes, OCB does not cause ciphertext expansion; it produces ciphertext the same length as the plaintext. The MiniSec authors port Rogaway's OCB implementation into 4000 lines of nesC code. Their implementation consumes 874 bytes of RAM and 16 KB of code memory.

MiniSec is composed of two schemes, MiniSec-U for unicast messaging, and MiniSec-B for broadcast messaging. The two schemes differ in the way they handle counters for replay protection. As in the SNEP protocol discussed previously, MiniSec-U requires each receiver to maintain a counter for each sender. MiniSec-B makes use of Bloom filters to defend against replay attacks.

MiniSec-U reduces the cost of transmitting counters. On the Telos platform, radio transmissions consume the most energy. Sending a single byte consumes as much energy as executing about 4,720 instructions. TinySec conserves radio energy by only sending the last few bits (the LB value) of the 64-bit counter. A developer can select an LB value based on the potential for dropped packets. A low LB value can be used in environments with less potential for interference, thus reducing the communication overhead. The protocol requires memory to store two keys and two counters for each pair of communicating nodes, one each per direction.

Counters cannot be used in broadcast communication because of the complexity of keeping the counters synchronized among multiple nodes. OCB provides confidentiality and authentication for broadcast communication, but MiniSec requires a novel approach to defend against replay attacks. One proposal recommends use of a sliding window. The

tactic splits time into a series of finite epochs. Estimated network latency dictates the duration of each epoch. A unique ID assigned to each epoch serves as the nonce. When a node receives an encrypted message, it deciphers the message with both the current epoch ID and the previous epoch ID. If neither ID yields successful decipherment, the node flags the packet as a potential replay.

With this approach, a packet sent early in an epoch can be replayed throughout the epoch. Bloom filters and loose time synchronization help defend against this replay attack. Bloom filters allow nodes to efficiently store a fingerprint of received messages into an array and quickly query the array to determine if the message has already been seen. Bloom filters guarantee that replayed messages will be detected. However, this strategy may flag some legitimate new messages as replayed messages. Bloom filters detect replayed messages within an epoch. Each receiving node maintains two Bloom filters: one for the current epoch and one for the previous epoch. Upon receipt of a packet, the recipient first validates the packet by performing OCB decryption. If this succeeds, the recipient performs a test to determine if the packet was replayed. The receiver first queries the Bloom filter for the packet. If the receiver finds the packet in the Bloom filter, it considers the packet a replay. If the receiver does not find the packet, it considers the packet fresh and adds it to the Bloom filter. This strategy will detect all replay attacks within the epoch.

MiniSec provides a high level of confidentiality and integrity with less overhead than its predecessors do. Implementation details such as packet length, key length, and the authentication tag influence the level of security. MiniSec shares the following fields with

TinyOS: length, frame control, sequence number, destination PAN, destination address, Active Message (AM) type. . Like TinySec-AE, MiniSec adds a 2-byte source address and replaces the 2-byte CRC with a 4-byte tag. Use of cryptographic keys replaces the functionality of the TinyOS group ID, thus this field is dropped. Overall, MiniSec adds three bytes to the standard TinyOS packet. Providing both authentication and encryption, MiniSec consumes one-third the energy of TinySec-AE.

Like TinySec, MiniSec employs Skipjack block size of 64 bits as the underlying block cipher. OCB requires the nonce to be the same length as the block size, so MiniSec uses a 64-bit counter. This monotonically increasing counter guarantees semantic security. MiniSec follows the Skipjack standard and requires 80-bit keys.

MiniSec achieves notable efficiency while maintaining a high level of security. It reduces the transmission overhead of TinySec-AE by two bytes and it reduces computational overhead by employing OCB. This strategy elevates MiniSec as one of the most efficient algorithms to provide confidentiality and integrity to wireless sensor networks.

5.6 AMSecure

While TinySec and MiniSec evaluated the cost of software-based cryptography, the AMSecure [37] poster abstract describes implementation of hardware-accelerated cryptography in TinyOS. They insert a cryptographic acceleration hardware module wired between the Active Message (AM) module and the radio. When transmitting or receiving

an AMSecure message, the AMSecure module performs the cryptographic operations, and then sends the payload to the other layers.

AMSecure offers the four cryptographic modes specified in IEEE 802.15.4 [38]: no cryptography, authentication-only with CBC-MAC, encryption-only with CTR mode, and authenticated encryption with CCM (CTR with CBC-MAC). The system appropriates between one and nine bytes of the 29-byte TinyOS payload, depending on the cryptography utilized. Unlike its predecessors, AMSecure bases all cryptographic operations on the AES block cipher. Like TinySec, AMSecure preserves backward-compatibility with legacy TinyOS application by controlling activation of cryptography with a compile-time flag. This flag distinguishes standard TinyOS Active Messages from AMSecure messages.

The poster abstract summarizes the processing and communication overhead of AMSecure. By using a hardware cryptographic module, AMSecure message processing time is kept to a low, predictable level. Since it offloads cryptographic operations, the overhead decreases as payload length increases. With a standard 29-byte payload, addition of authentication and encryption result in 20 percent message overhead. This drops to approximately 8 percent with a 100-byte payload. For a 90-byte payload, the receive processing time increases from about 500 microseconds for a standard TinyOS message to 1750 microseconds for an authenticated, encrypted AMSecure message. This represents a significant improvement over the 50 microseconds per byte rate documented by Law and Choi [31, 32].

5.7 SecureSense

SecureSense [39] dynamically enables a sensor node to modify its security controls based upon observations about the external environment and requirements from the application. The approach differs from the previous security protocols where the sensor must apply one level of security to all messages at all times. Dynamic variation of security controls allows sensors to preserve precious power resources when the threat level is low. SecureSense aims to provide confidentiality, integrity, access control, semantic security, and message replay protection.

Not all sensor network applications require strong adherence to the goals of information security. For example, in a common reference grid application sensors are deployed as a local version of GPS. A receiver uses position reports provided by the sensors to determine its location. Since there are multiple, redundant sensors, availability ranks as a low priority. Since the sensors only provide location data, the network can overlook confidentiality. The threat of message spoofing, however, demands consistent protection with message authentication. Other applications may require periodic utilization of all security goals at some point in the sensor network lifetime. SecureSense allows provisioning enough security to balance risks and countermeasures.

SecureSense acts as a runtime security service in the TinyOS radio stack. This TinyOS stack includes five layers: application, Active Message, radio packet, radio byte, and RF module. The radio byte component sends and receives bits one-by-one over the RF module. The radio packet module spools incoming bytes to recompose them into packets. It is responsible for tagging packets destined for specific application level components.

SecureSense inserts a security broker between the radio packet and the radio byte components. It replaces the 8-bit Active Message field in the TinyOS packet header with a SecureSense Security Composition ID (SCID).

The security broker calls upon service modules in a service library. These services interface with cryptographic functions to fulfill requirements identified in the SCID. The first four bits in the SCID identify the available security capabilities: confidentiality, integrity, semantic security, and replay protection. The final two bits identify cipher strength. Two other bits remain unused. Depending on the value of the SCID, other fields in the TinyOS packet header may not be necessary. For example, a group cryptographic key implicitly identifies group membership, thus the 1-byte group ID has been removed. When a message requires only error detection, SecureSense fills the last two bytes with a CRC. When the threat landscape demands message integrity and authentication, a message authentication code (MAC) replaces the CRC.

While SecureSense embraces dynamism, it requires installation of the components of the security service library prior to deployment. Thus, the range of security options depends on the constraints of the hardware platform. Efficient, reusable code improves configuration options. SecureSense utilizes the RC5 block cipher to enable this efficiency. In RC5, key size, block size, and number of rounds can be defined at runtime depending on the security strength outlined in the SCID. The authors evaluated SecureSense using the optimized RC5 implementation from TinySec, which preserves resources better than the SPINS implementation or the default C implementation.

The evaluation shows that SecureSense can conserve power by balancing security capabilities with the threat environment. This evaluation specifically measures energy required for communication and omits evaluation of energy consumed by computation. When SecureSense provides confidentiality, integrity, semantic security, and replay protection, SecureSense consumes an equivalent amount of communication energy to TinySec. Since SecureSense truncates the CRC field, it actually introduces less communication overhead in insecure mode than standard TinyOS.

5.8 Interleaved Authentication

As illustrated in section 4.3, wireless sensor networks face significant exposure to node compromise. The compromise could be as simple as physical destruction of a node or sophisticated enough to allow an attacker to manipulate an active, compromised node. An adversay who overtakes a legitimate, active node may use this node to inject false information into the system. Secure key rotation alone cannot defend against this threat. Compromising an active node gives the attacker access to keying material, thus the attacker has the ability to calculate legitimate message authentication codes. These reputedly authentic messages can misguide the base station, divert intrusion alerts, and deplete network resources.

When analyzing security of wireless sensor network protocols, the designers of [40] assume that nodes will be compromised. Zhu, et al. present an interleaved hop-by-hop authentication method that can detect such false data injection attacks. The method adds an additional layer of security on top of SPINS, TinySec, and others by defining a threshold for the number of compromised nodes that a network can endure. They define a value *t*

representing the maximum number of compromised nodes tolerated during an attack, and require *t + 1* nodes to send an authenticated reports of an event. The scheme guarantees that if no more than *t* nodes are compromised, the network will detect and drop falsely injected data. A designer can adjust the value *t* based on the threat of node compromise. The strategy proposed by Zhu, et al., defends against collusion among compromised nodes.

The proposal by Zhu, et al., requires a network topology organized into clusters, with a subset of nodes acting as cluster heads. The cluster must include at least *t + 1* nodes, including the cluster head. The cluster may reside multiple hops from the base station. Nodes within the network can send unicast messages up and down the tree, and broadcast messages to their neighbors. Nodes share a master key with the base station and have the ability to establish shared keys with most of their one-hop neighbors.

The scheme relies on an association of nodes that reside *t + 1* hops apart on the path to the base station. They refer to the peer closest to the base station as the “upper associated node” and the lower peer as “the lower associated node”. Upper associated nodes validate the message authentication code (MAC) appended to messages from their lower associated peers. Each message may carry as many as *t + 1* MACs as it travels from leaf nodes to the base station. A validation failure on any of the MACs will cause the message to be dropped. As long as the number of compromised nodes remains below the value *t*, the system can detect false data injection.

Five unique phases comprise the hop-by-hop authentication technique, including initialization and deployment, association discovery, report endorsement, en-route filtering, and base-station verification.

During node initialization and deployment, the key server loads each node with a unique id and a unique key that node shares with the base station. The node derives an authentication key from the encryption key it shares with the base station. The key server then can use one of many key establishment algorithms to initiate network key distribution. This enables nodes to establish shared keys with their neighbors.

The association discovery phase allows nodes to discover their associated peers both on the path downward from the base station and in reverse. The base station kicks off the process by broadcasting a hello message. Each node that receives the broadcast checks for the id of the node $t + 1$ hops up the tree, replaces that id with its own, and rebroadcasts the modified hello. This provides an upper bound of $t + 1$ node ids attached to the hello message. A receiving node records the id of the node $t + 1$ hops up the tree as its upper associated node. Note that nodes less than $t + 1$ hops from the base station do not have an upper associated node. When the hello message reaches a cluster, the cluster head assigns its leaves to upper associated nodes. The hello message can be authenticated with a broadcast authentication scheme such as μTESLA.

After the cluster notifies its leaves of their peers, it sends an acknowledgment back to the base station. The lower associated nodes authenticate the acknowledgment with the pairwise key they share with their upper associated node. Along with the MAC, the

acknowledgment includes the node ids of the cluster head and the leaf nodes. As the acknowledgment is returned up the tree toward the base station, upper associated nodes learn the node id of their lower associated node. They replace the id of their lower associated node with their own id and forward the acknowledgment back up the tree. In cases where upper nodes have branches to multiple clusters, they record cluster ids and nodes in a table.

When nodes witness an event, they send a report to the base station. This hop-by-hop proposal requires $t + 1$ nodes to witness an event and endorse a report of the event. As the endorsement moves upstream, the ids of the nodes that witnessed the events and authentication codes will be appended to it. Each node computes two MACs for the event. The individual MAC is computed using the node's key with the base station. The pairwise MAC is computed using the node's pairwise key with its upper associated node. If the cluster head can authenticate a report from all its leaf nodes, it compresses the individual MACs by XORing its individual MAC with the individual MACs from the leaf nodes. The pairwise MACs are not compressed.

Upper level nodes that receive this report must authenticate it using their pairwise key with their lower associated node. If authentication succeeds, the node will extract the MAC from its lower associated node and append its own. If authentication fails, the message will be dropped. This in-route filtering assures that as long as no more than t nodes are compromised, then falsely injected data will be dropped.

Once the alert reaches the base station, the base station performs its own verification. It extracts the event data and node ids from the report. It then computes its own compressed MAC on the event data using its keys shared with the nodes in the node list. If the MAC matches the compressed MAC in the report, the alert is considered valid.

Since wireless sensor network are susceptible to damage, the Zhu proposal includes maintenance techniques. One strategy proposes piggybacking association discovery messages on base station beacons such as those sent in TinyOS. Nodes accept the first beacon they receive as their parent node. Since these beacons are sent every epoch, it is possible for nodes to change parents every epoch. While this strategy is satisfactory for dynamic networks, it is costly for networks that do no change frequently. A less costly base station initiated strategy has the parent change only if the node determines that any of the nodes in the beacon from its parent have changed. Repair can also be initiated locally when nodes detect failure of a neighbor. The proposed technique requires nodes to use GPS or similar technology to determine the physical location of their neighbors. When a node detects that its parent has failed, it will send a REPAIR message to the first node counterclockwise from the edge between itself and its deceased parent. It will then exchange messages with this node to learn the ids of the upstream nodes and rebuild any broken node associations.

Zhu's proposal provides a higher level of security than simple pairwise authentication between neighboring nodes. The base station can trust that reports from leaf nodes are authentic based on the MAC computed with the pairwise key between itself and

the leaf node. Intermediary nodes can authenticate reports based on pairwise keys with their lower associated nodes. As long as $t + 1$ nodes agree on an event, the system will detect a falsely injected report sent by t nodes or less. This high level of security requires a high level of node redundancy.

5.9 Comparison of Implementations

This thesis aims to integrate into OCO security countermeasures that balance the cost of security with the risks of a target tracking system. Superficially, an authentication-only solution like TinySec-Auth appears to adequately secure the system without significantly affecting network longevity. However, all of the solutions deserve a comparative analysis. Many of the proposals in this section evaluate overhead caused by communication costs, processing costs, and speed of cryptographic operations. This subsection will compare the costs of these proposals side-by-side. Changes in mote technology make this analysis complex since the different radios and processors exhibit different power consumption behavior. Table 4 summarizes the platforms on which the preceding proposals were evaluated.

Table 4: Summary of Mote Platforms Used by the Various Methods

Proposal	Platform	Processor	Speed	Architecture	Radio Hardware
SNEP	Smart Dust	Unknown	4 MHz	8-bit	916 MHz,
RPT, LEA	Moteiv Telos	TI MSP430	8 MHz	16-bit	Unknown
TinySec	Mica, Mica2, Mica2dot	Atmel ATMega128(L)	8 MHz	8-bit	RFM TR1000, Chipcon CC1000
MiniSec	Moteiv Telos	TI MSP430	8 MHz	16-bit	Chipcon CC2420
AMSecure	MicaZ	Atmel ATMega128(L)	8 MHz	8-bit	Chipcon CC2420

Some protocols immediately stand out for their high communication overhead. The additional 8-byte MAC in the SPINS method SNEP, for example, clearly exceeds the

communication overhead required by its successors such as TinySec and MiniSec. This large MAC significantly reduces the chance of message forgery, but other proposals agree that the excessive length provides too much security for low bandwidth wireless sensor networks. The RPT and LEA proposals seem appealing because they both address specific communication patterns common to OCO: RPT can secure the regular and predictable health messages sent between parents and descendants; LEA can effectively protect the low entropy intrusion alerts sent from a border node to its parent. However, their proposal does not specify integration of RPT and LEA into one implementation. It requires different solutions for the different communication patterns in OCO. As a final drawback, the RPT and LEA proposals do not include analysis of the energy consumption characteristics of the protocols. A more current protocol, AMSecure seems promising because it advocates use of new standards like IEEE 802.15.4 and the NIST approved block cipher AES. With hardware-supported cryptography, AMSecure can quickly encrypt a packet in 1750 microseconds. However, the short AMSecure poster abstract has yet to be fully documented. Table 5 illustrates the communication overhead required by proposals that integrate security into a standard TinyOS packet. The last column shows the portion of total packet overhead dedicated to security.

Table 5: Increase in Packet Length

Proposal	Payload (b)	Total Overhead (b)	Total Size (b)	Security Overhead (b)
TinyOS	24	12	36	0
SNEP (SPINS)	24	18	42	6
TinySec-Auth	24	13	37	1
TinySec-AE	24	17	41	5
MiniSec	24	15	39	3

This leaves TinySec and MiniSec as the two most promising, low overhead contenders. Both employ well-documented, standards-based cryptographic block ciphers: CBC-MAC and OCB. Both proposals make available the nesC source code used in their evaluation. One can compare the costs of the two proposals simply by evaluating the increase in packet length. While authentication-only TinySec-Auth increases packets by 1-byte, the authenticated-encryption TinySec-AE and MiniSec increase packet length by 5-bytes and 3-bytes respectively. These increases take a direct toll on the energy resource since the radio must be turned on longer when transmitting or receiving longer packets. The cost of sending a packet involves more than just the transmission of bits of data and a header. Longer packets also increase communication latency and consume valuable processor cycles.

Table 6: Energy Consumption of TinySec and MiniSec

Evaluated Protocol	Implemented Security Options	Energy Consumption (mAs)	Increase in Consumed Energy
TinySec	TinyOS	0.5760	-
TinySec	TinySec-Auth	0.5940	3 %
TinySec	TinySec-AE	0.6336	10 %
MiniSec	TinyOS	0.0340	-
MiniSec	TinySec-AE	0.0387	13.9 %
MiniSec	SNEP	0.0415	22.2 %
MiniSec	MiniSec	0.0368	8.3 %

The TinySec and MiniSec authors both empirically evaluated the communication cost associated with their respective proposal. Table 6 summarizes the relative costs of the two methods with different security options being implemented. Three security options were implemented in TinySec, including TinyOS, TinySec-Auth, and TinySec-AE [8]. Four security options were implemented in MiniSec, including TinyOS, SNEP, and

TinySec-AE, and MiniSec [34]. The TinySec authors [8] evaluated energy consumed by an 8-bit processor when transmitting packets through a Chipcon CC1000 radio. They sampled current drawn by a transmitter when sending a packet with a 24-byte payload. The MiniSec authors [34] selected a more contemporary mote model, with a 16-bit processor and the Chipcon CC2420. While their actual energy consumption measurements differ significantly from those in TinySec, the percent increase appears similar. TinySec-Auth clearly keeps the cost of authentication lower than its authenticated encryption counterparts do.

6 Problem Statement

When used for detecting intrusions and tracking targets, wireless sensor networks must provide a high assurance of trust. Modification or fabrication of alerts introduces false positive conditions that reduce the trustworthiness of the network. Forgery of routing messages may enable an attacker to occupy key roles within the network, allowing the attacker to control the flow of information toward the base station. If a target tracking wireless sensor network is deployed in an arena where it is subject to these attacks, the value of its service will be diminished. False alarms lead to operational inefficiencies in any environment. In defense applications, manipulation of alarms may lead to loss of valuable assets or even loss of life. A layer of authentication wrapped around wireless sensor networks mitigates these risks.

This thesis summarized generic attacks against wireless sensor networks in Section 4.3 and ranked threats against OCO in Section 4.4. These sections illustrate the ability of an attacker to selectively dismantle the target tracking and alert notification service provided by the network. The analysis demonstrates that the efficiency improvements in OCO can be circumvented, transforming OCO into an inefficient configuration. The attacks against OCO plague other target tracking applications such as Direct Communication, LEACH, and GDAT. All three share an application layer alerting process vulnerable to message forgery. LEACH, GDAT, and OCO possess self-organization capabilities that expose

vulnerable network routing services. As shown in the risk assessment (section 4), both the network organization and the intrusion notification functions need protection. A brief survey of related literature shows that there has been little convergence of security protocols with target tracking applications. Perhaps this originates from the conflict between the increased costs of security measures with the network longevity goals of target tracking networks. Each of the security mechanisms surveyed in this thesis requires a trade-off between functionality, efficiency, and ease of deployment with protection of the network's assets. The remainder of this thesis proposes an efficient security countermeasure for OCO and evaluates the cost of this countermeasure in terms of energy efficiency.

7 Proposed Solution: s(OCO)

The OCO risk assessment demonstrated a significant need for origin integrity and message integrity. Focusing strictly on authentication, instead of confidentiality or availability, balances the risk outlined in the risk assessment with the goal of conserving energy. Of the security proposals reviewed in this thesis, TinySec-Auth provides the appropriate level of security. TinySec-Auth increases communication overhead by only 1-byte per packet. While TinySec's use of weak keyed RC5 and Skipjack ciphers poses some security risks, their efficiency prevails until stronger ciphers such as AES become available for wireless sensor platforms. Other proposals, such as TinySec-AE and MiniSec, package both integrity and confidentiality into their solution. These consume additional computation and communication resources. While the proposal in [40] guarantees a measureable level of integrity in the presence of compromised nodes, it requires a clustered topology not available in OCO.

This thesis recommends integrating origin authentication and message integrity into any OCO message with a total risk rating above 250. This captures all messages in the Processing, Tracking, and Maintenance phases. This authenticated version of OCO, known as s(OCO), provides individual message authentication with limited overhead. s(OCO) will protect the network from message fabrication and message spoofing as long as no nodes are compromised. If an attacker can compromise an active node, it can steal the shared key

and defeat the security protocol. Table 7 summarizes the 14 OCO messages, their respective roles, and their packet length.

Table 7: s(OCO) Packet Length

Message	Purpose	Packet Length (B)
M1	The sender seeks child nodes for adoption	7
M2	Nodes acknowledge their adoption with message M2	13
M3	The base informs child nodes of the id of their parent	16
M4	The base informs parent nodes of the id of their descendants	16
M5	The base informs border nodes of their occupation	14
M6	Assign redundant nodes to a sleep state	18
M7	Inform routing nodes of their occupation	14
M8	A sensing node alerts the base station of an intrusion	18
M9	A sensing node alerts its neighbors of an intrusion	12
M10	A child node reports its health to its parent	16
M11	A parent advertises its health to descendants	12
M12	A child reports that is has lost its parent	12
M13	A parent reports that is has lost its child	18
M14	The base resynchronizes redundant nodes	16

In order to model the system, this proposal imposes a standard TinyOS packet format onto OCO communications and establishes standard sizes for OCO data fields. This facilitates calculation of the cost of integrating TinySec-Auth into OCO. Common fields among packets include destination address, Active Message (AM) type, and packet length. By starting packets with the destination address, nodes may employ *early rejection* of messages. When a node determines that it is not the intended recipient, it may conserve energy by dropping the packet. The active message type, analogous to a TCP or UDP port in the Internet protocols, specifies the appropriate handler function to extract and interpret the message on the receiver. s(OCO) maps message type (M1 … M14) into the TinyOS Active Message type field.

Figure 8 illustrates the respective packet formats for TinyOS, TinySec-Auth, and s(OCO). Shaded fields in the packet diagrams represent fields protected by the MAC.

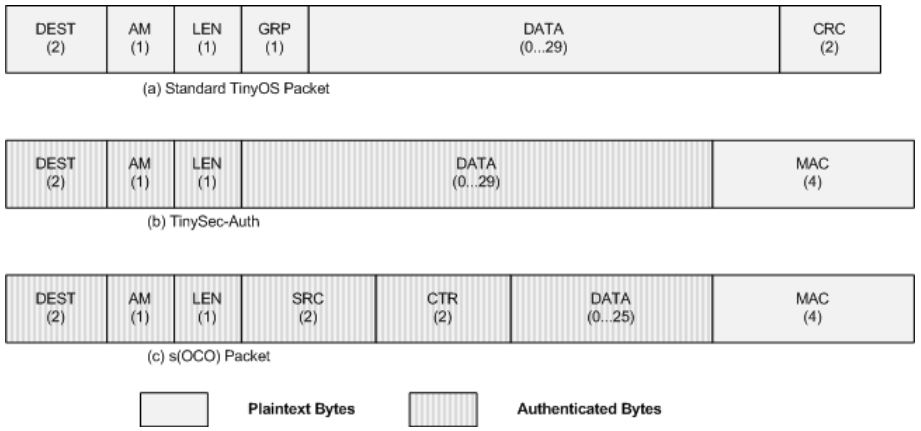

Figure 8: Packet Formats

s(OCO) follows the packet format in TinySec-Auth and increases the TinyOS headers by one byte. Both proposals drop the 1-byte group ID and the 2-byte CRC fields in the original TinyOS packet and replace them with a 4-byte MAC. The MAC provides the packet integrity service of the CRC. The cipher key implicitly replaces the group membership function provided by the group ID. s(OCO) appropriates bytes from the payload for additional fields including a counter used as a message id and the packet source address. Node addresses occupy two bytes. s(OCO) allocates 2-bytes each for time-to-synchronize and time-to-stay-awake. Node position, node energy level, and notification timestamps each receive 4 bytes. The standard fields in a s(OCO) packet consume 12 bytes.

7.1.1 s(OCO) Position Collection

The Position Collection phase, which only occurs during network initialization, includes two messages with risks ratings below 250. The base broadcasts message M1, the

Position Request message, immediately following node deployment. The nodes respond by sending message M2 to their parent, which in turn forwards the message toward the base. Because of the narrow attack window, the Position Collection phase messages receive a low total risk rating. Thus, implementation in a standard TinyOS packet format satisfies security requirements. Message M1 occupies 7 bytes and maps to TinyOS Active Message (AM) type 1. The Position Reply message, message M2, includes fields for reporting node ID and that node's position. These increase packet length of M2 to 13 bytes. Figure 9 shows the composition of messages M1 and M2.

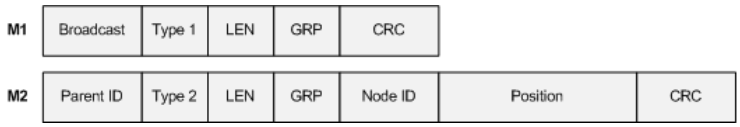

Figure 9: Position Collection Packet Format

The Processing, Tracking, and Maintenance phases expose the network to a higher risk. Nodes accept Processing, Tracking, and Maintenance phase messages throughout the network lifetime. Thus, s(OCO) uses TinySec-Auth to secure the message contents.

7.1.2 s(OCO) Processing Phase

In the Processing phase, the base station sends two topology type packets and three packets used to assign roles to nodes. s(OCO) must broadcast the topology messages because, at this point in the network setup, there is no route from the base station to the destination nodes. The topology information captured during the Position Collection phase only provided the path for nodes to report their id and position to the base station. Message M3 advises a child node of the id of its parent. M4 informs a parent node of the id of one of

its children. A parent receives M4 for every one of its immediate children. The packets put the child node id and parent node id into the message payload, adding 4 bytes and increasing the length of M3 and M4 to 16 bytes, as shown in Figure 10.

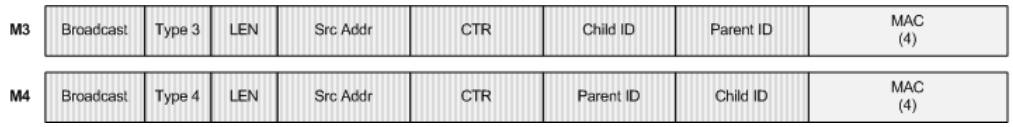

Figure 10: Processing Packet Format – Topology Messages

The OCO route propagation method raises an interesting dilemma. Parent nodes only know about their immediate descendants. Unlike other target tracking methods, OCO does not require a forwarding node to have a specific number of children or members in a cluster. Many parent nodes may have only one immediate child, yet through that child, they may have a large number of descendants. This organization allows an OCO network to spread a long distance from the base station and cover a large detection region. However, it makes communication from the base to the border costly. Since a parent node's lineage may span through many generations of descendants, a resource constrained parent node cannot be expected to have enough memory to store the id of all its descendants. Thus, OCO lacks a route from the base station to the perimeter and messages from the base to the border must be broadcast.

The remaining three messages in the Processing phase assign node occupation. Because of the lack of a route from the base to the border, the base must also broadcast these messages. When a node receives on of these messages, it will check the id of the intended target in the payload and rebroadcast the message if necessary. Nodes use the

counter to track whether or not they have already broadcast the message. M5, which requires 14 bytes, instructs a border node to activate its tracking sensor and its radio. M6 announces the time to sleep (TTS) and time to stay awake (TTSA) to redundant nodes. These time fields consume two bytes each and increase packet length to 18 bytes. Message type 7 consumes 14 bytes to instruct forwarding nodes of their occupation. Figure 11 shows the packet format for message 5-7.

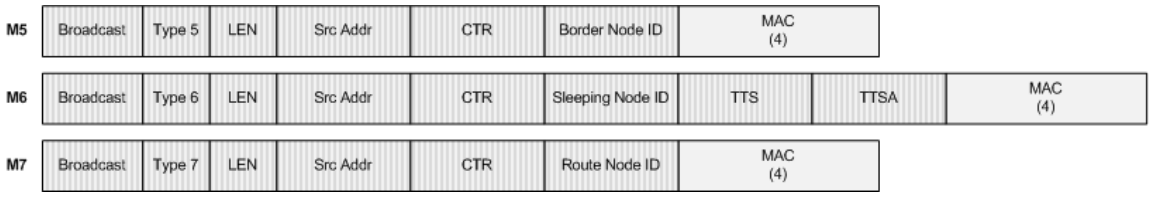

Figure 11: Processing Packet Format – Occupation Messages

7.1.3 s(OCO) Target Tracking

The two messages in the Target Tracking phase originate from a border node alerting its peers of an intruder. M8, sent toward to base station, includes fields for reporting node id and a 4-byte timestamp. It occupies 18 bytes in a TinySec-Auth format. A node broadcasts M9, which requires 12 bytes, to its neighbors to inform them of the intrusion. Figure 12 illustrates the format of the Tracking phase messages.

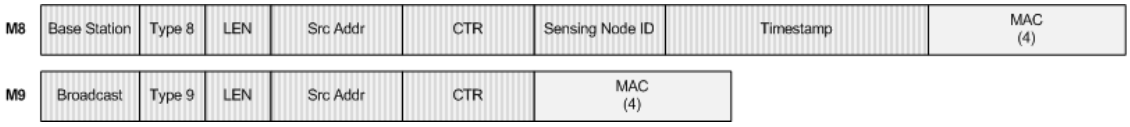

Figure 12: Tracking Phase Packet Format

7.1.4 s(OCO) Maintenance Phase

The Maintenance phase supports network longevity with keep-alive messages and notifications when nodes lose their parent or child. Messages M10 through M14 constitute the Maintenance phase. By way of message M10, a child node can report its health to its parent. M10 includes a 4-byte field where the child node records its energy level, increasing total packet length to 16 bytes. A parent informs its children that it is still alive by broadcasting M11, which requires 12 bytes. Nodes that receive M11 do not rebroadcast it, as they would when they receive one of the Processing phase messages. However, nodes that receive the message must still authenticate it to determine if the source address belongs to their parent. s(OCO) does not define recommended timing interval for sending M10 and M11, leaving a tradeoff between recovery time and energy use to the implementation. Figure 13 shows the organization of messages M10 through M14.

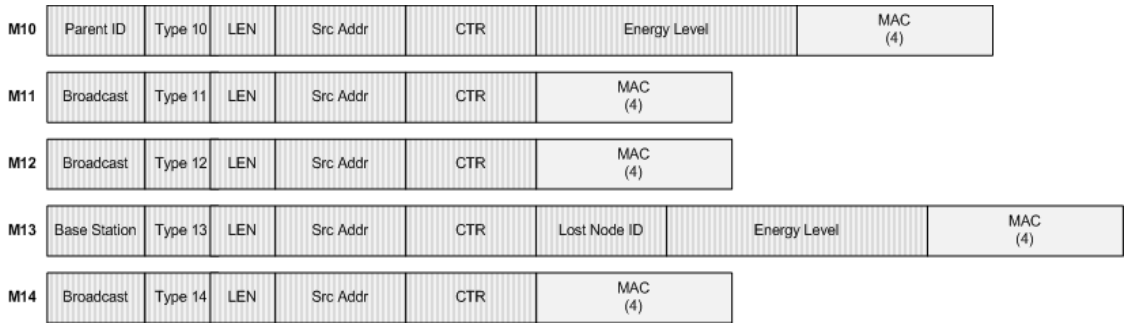

Figure 13: Maintenance Phase Packet Format

Message M12 and M13 make up the S.O.S. messages in s(OCO). A child node broadcasts the 12-byte message M12 when it does not receive message M11 from its parent. Neighboring nodes must authenticate, but not rebroadcast M11. A parent sends M13 to the base station when its child node fails to report its status. M13 includes 2-bytes

for the lost child node id and 4 bytes for the parent node's energy level, lengthening it to 18 bytes. Each node that receives M13 must authenticate it and send it to their parent until it reaches the base. The base station periodically sends message M14 to resynchronize redundant nodes. This message includes updates to the "time to synchronize" and "time to stay awake" parameters. M14 consumes 16 byes.

8 Experimental Design

The thesis puts forth the hypothesis that securing OCO will increase the total cost of operating the network to between three percent and thirteen percent. The three percent lower bound reflects the cost of a packet in TinySec-Auth with a full 24-byte payload. The thirteen percent upper bound represents the cost increase of s(OCO)'s shortest 12-byte packets. The mean operating cost of s(OCO) should exist within these upper and lower bounds because of packet length and the influence of the sensor module and the radio module. The experiments will simulate an OCO network and an s(OCO) network and evaluate the mean operating costs of both networks under similar circumstances.

8.1 Simulation Tools

The experimental analysis employs the OMNeT++ [4] network simulator for the implementation and evaluation of the s(OCO) countermeasures on network life span. OMNeT++, a public source object-oriented simulation tool, provides a framework that simplifies evaluation of communication protocols. OMNeT++ supplies a hierarchal set of modules, each interconnected through interfaces called *gates*. Since OMNeT++ manages transmission of messages through the gates, the developer can focus on implementation of application classes within each module. In this evaluation of OCO, an instantiation of an OMNeT++ application class randomly distributes nodes across the simulation grid during the Position Collection phase. It simulates the transmission and reception of Position Collection messages and tracks the cost of each message throughout the simulation. A

separate C# application reads the output from OMNeT++, constructs the coverage map, and performs the image processing tasks. This application determines node occupation, and organizes the network topology. The output from this application is fed back into the OMNeT++ simulator to evaluate the cost of message passing in the Processing and Tracking phases. As in [12], the simulation omits modeling of the Maintenance phase. The OMNeT++ simulator and the C# image processing application lack automated interfaces that could allow simple integration of the two components. Without such interfaces, the network cannot seamlessly notify the base of the need for maintenance, reprocess the coverage map, and send new topology and occupation messages.

Since this thesis concentrates on efficient energy management, the simulation measures energy consumption and counts the number of messages sent and received by each node. The simulation assesses energy consumed by the node's radio, its sensor, and its microcontroller. In the Position Collection and Processing phases, all nodes maintain an active radio and processor. Thus, a node's energy consumption in these first two phases depends mainly on the number of messages it has to send and receive. In the Tracking phase, a node's occupation influences its energy usage characteristics. Border nodes generally consume the most energy because both their sensor modules and radio modules remain active. Their processor sleeps until it is required to create a message. Forwarding nodes should consume less since they keep their sensor disabled until one of their neighbors detects an intruder. Their radio remains enabled to receive and forward messages. As with border nodes, their microcontroller sleeps except to create messages. All three components of redundant nodes remain deactivated, although they periodically wake

up to receive commands sent by the base. The simulation assumes that the base station has unlimited energy and computation resources.

The application code in this simulation is derived from the original OCO simulation developed in [1]. This simulation requires two major modifications to the original code. Updates to each application class in the simulation reflect the cost of both TinyOS and TinySec-Auth messages. The applications prompt the user to select the security mode upon application startup. A new application was created to simulate message passing in the Processing phase. Counters were added to each application to track the number of messages sent and received.

8.2 Simulation Model

A module referred to as the *system* resides at the top level of any OMNeT++ hierarchy. The model developed for OCO and s(OCO) includes three sub-modules: a manager, an intruder object, and a sensor node object. The manager module orchestrates the simulated communication among all nodes. It tracks the position of objects in the model and simulates connections between objects whenever they are within radio transmission range or sensor detection range. For example, when the manager determines that an intruder object is within the sensing radius of a node, it forms a connection between the intruder node and the sensor node.

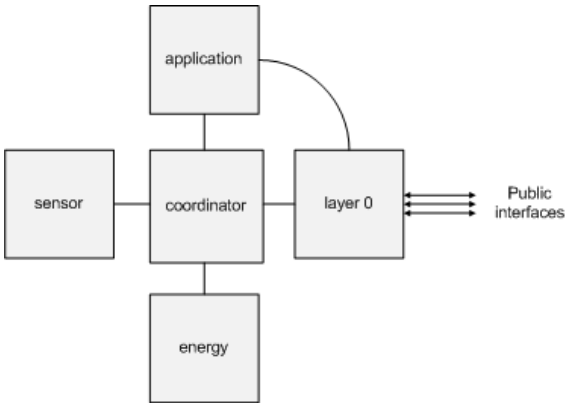

Figure 14: OMNeT++ Sensor Node

The sensor node object encapsulates simple modules representing a sensing device, a power storage device, a network interface, and a controller. Figure 14 illustrates the organization of a sensor node and the connections among modules within the node. The application component contains the main methods that define an OCO node. The definition of each of the 14 OCO messages can be found in the application module code. The application code also includes timers that deduct power on nodes with active processors or sensor components. The coordinator module links the application to a node's internal components, the sensor module, the energy module, and the radio module. The application module deducts energy during use of the processor, radio, or sensor by sending messages through the coordinator module to the energy module. The layer 0 module, which simulates the network interface, connects one node to another. The OCO simulation code calculates a node's position and proximity to neighboring nodes. When nodes reside within one another's communication radius, the simulator connects the gates of their layer 0 modules. When the application simulates transmission of a message, OMNeT++ pushes the message to all nodes with layer 0 connections to the sender. Similarly, when the intruder object

comes within an OCO node's sensing radius, the manager connects the intruder with the OCO node through their layer 0 modules. The OMNeT++ object-oriented architecture simplifies application modeling and allows for visualization of network communications.

8.3 The Simulation Process

The evaluation of the OCO and s(OCO) methods follow the same simulation process. The simulation breaks OCO into four independent software applications: Position_Collection, WSN_O_Track, Processing, and Tracking. The first application simulates the Position Collection phase. In this application, the user defines the number of nodes, their communication radius, their sensing radius, and other network parameters. In each application, the user can elect to require TinySec-Auth or not. In this experiment, the Position Collection simulation runs without authentication. Once the user configures the simulation parameters, OMNeT++ distributes nodes throughout the plane as shown in Figure 15. The larger circle in the upper right quadrant represents the base station. OMNeT++ then simulates the Position Collection phase, sending messages M1 and M2 throughout the network. Figure 16 shows the shaded nodes near the base that have received message M1, flagged themselves as adopted, and sent message M2 to the base station. Once all nodes have been adopted, the simulator saves node id, position, and energy level in a flat text file. While OMNeT++ provides strong GUI support for simulation visualization, GUI simulations of large networks may take days to complete. Fortunately, OMNeT++ provides a command-line simulator that significantly reduces simulation time while producing precisely the same results.

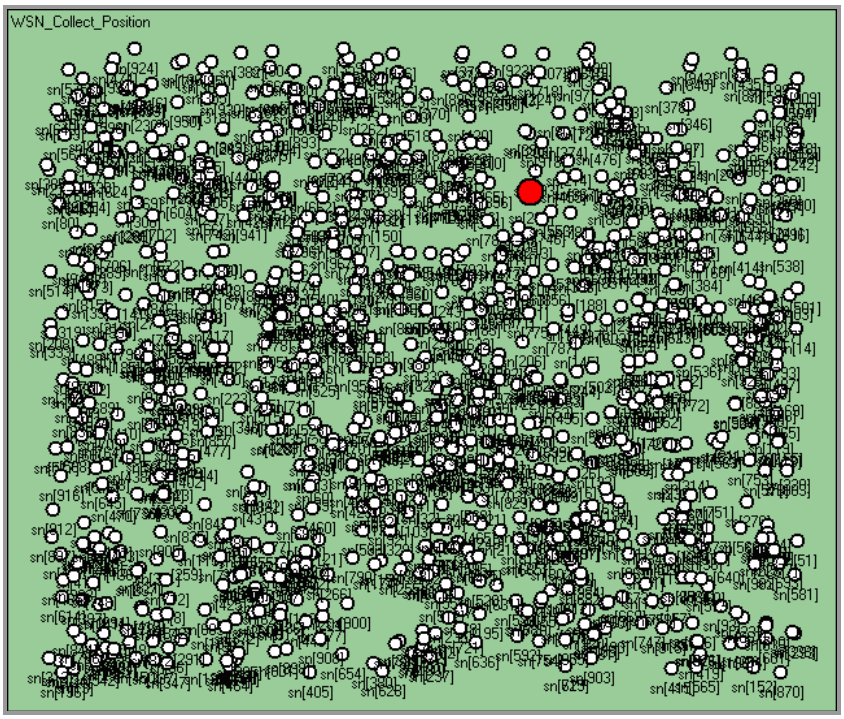

Figure 15: Start of Position Collection Phase – 1,000 Nodes

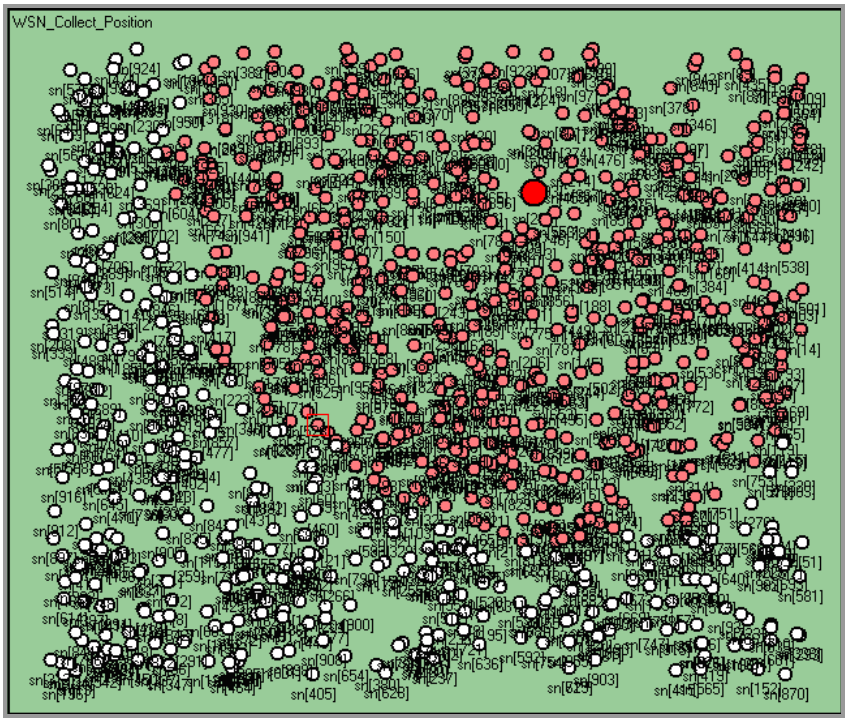

Figure 16: Position Collection in Progress – 1,000 Nodes

The second software application, WSN_O_Track, processes the image, determines the perimeter and node occupation, and defines the network topology. The user imports the flat text file from the Position_Collestion application and selects the menu option to process the image. Figure 17 shows a network of 1000 nodes prior to processing, with each circle representing the sensing radius of an individual node. In this scenario, 1000 nodes cover a detection region represented by 640x540 pixels. Following image processing, only 294 of the 1000 nodes remain activated. These nodes will serve as border nodes or forwarding nodes throughout the simulation. The application flags the remaining nodes as redundant nodes and omits them from the rest of the scenario.

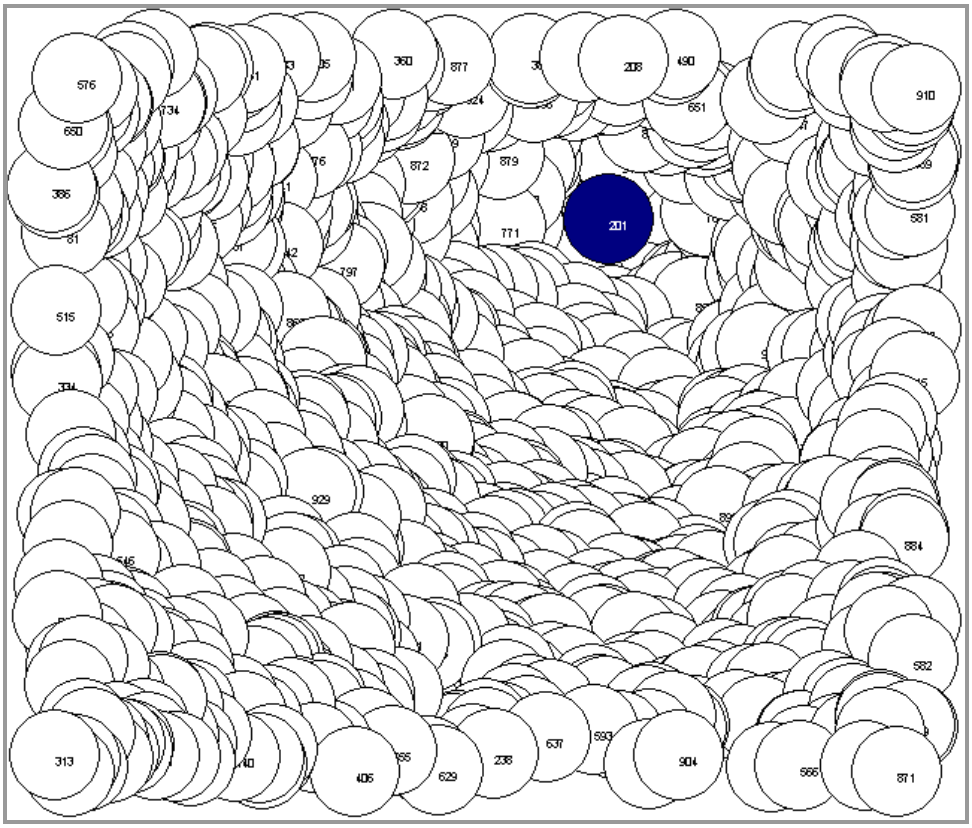

Figure 17: Start of Processing

Other menu options in the application enable visualization of the network topology. Figure 18 shows the region covered by the optimized set of nodes. In a high node density network such as the one in this example, the perimeter remains relatively smooth. A lower density network would produce a jagged border. Figure 19 illustrates the 60 nodes that have been selected to secure the border. Figure 20 outlines the path from the perimeter nodes to the base station. Figure 21 illustrates all network features. Although WSN_O_Track determines node occupation and network topology, it lacks the capability to simulate the cost of message passing. A separate OMNeT++ application addresses evaluation of message cost.

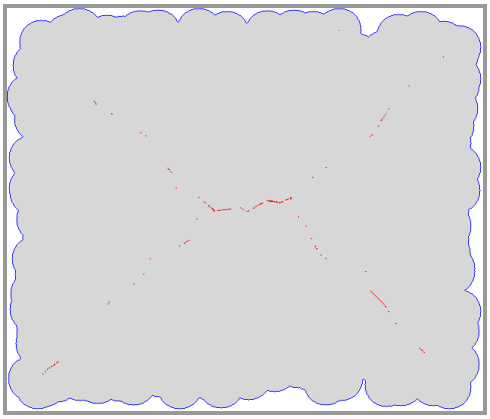

Figure 18: The OCO Perimeter

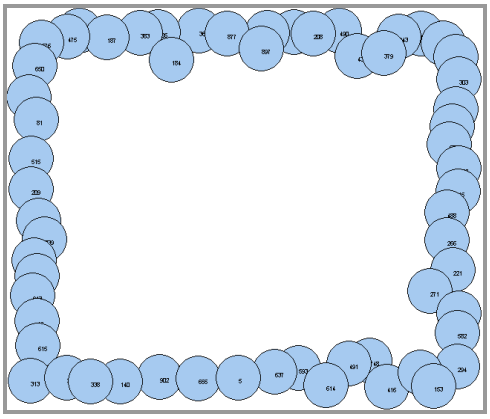

Figure 19: Border Nodes

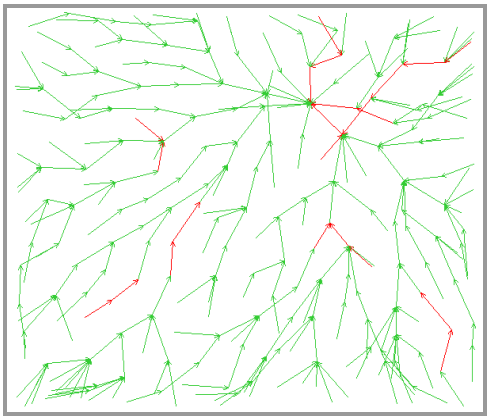

Figure 20: OCO Route Topology

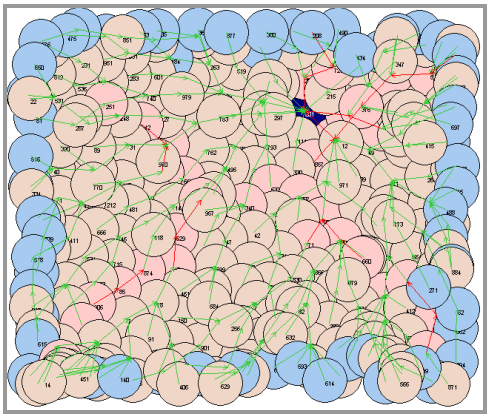

Figure 21: Processed Network

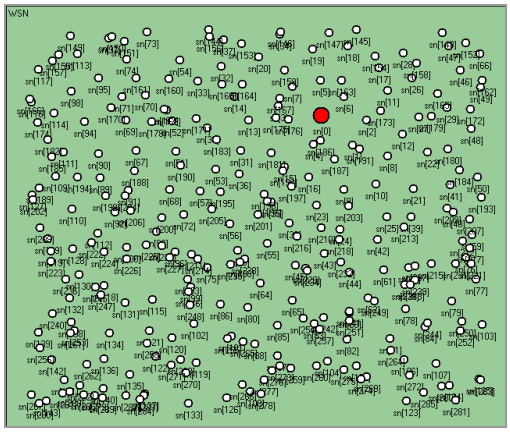

Figure 22: Processing in OMNeT++

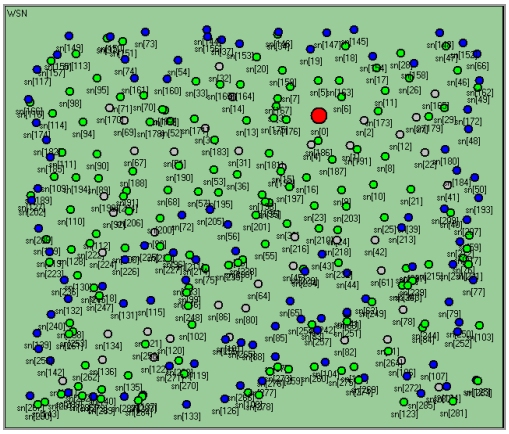

Figure 23: Completion of Processing

Once the C# image processing application completes the determination of node occupation and network topology, the OMNeT++ Processing application begins simulating the process of broadcasting the topology and role assignment messages throughout the network. This application reads the output from WSN_O_Track, draws nodes on the map, and updates their energy level. The application then begins sending messages M3 through M7, which define node occupation and network topology. Figure 22 and Figure 23 illustrate the start and conclusion of the Processing phase, respectively. Figure 22 shows nodes that have not been assigned an occupation filled in white. In Figure 23, dark blue nodes represent border nodes with activated sensors and radios. The lighter green nodes represent forwarding nodes that only maintain an active radio. Note that the screenshots in Figure 22 and Figure 23 represent an OCO network initialized with 500 nodes. In this less dense environment, interior nodes must fill the role of border node in order to guarantee a contiguous perimeter. When the simulator completes assignment of node occupation and definition of network topology, it records message count and node energy into output files.

The target tracking process starts with initialization of the Tracking application, which reads the output from the Processing application. The Tracking application instantiates an intruder object and nodes that make up the OCO network. The intruder object reads another input file that contains the coordinates and speed of its path through the coverage area. In the simulations, the intruder enters the OCO network penetrating the border and then travels through the interior. As the intruder enters the sensing radius of OCO border nodes, the manager module in the simulator will link the intruder with one or more OCO sensing nodes whose range covers the intruder's current location. The sensing nodes will alert the base station and their neighbors by sending messages M8 and M9. Neighboring nodes that have been alerted to the intrusion will change to a light coral color, which signifies that they have enabled their sensor. Figure 24 shows a small dot that represents the intruder near the left-center of the grid. As the intruder passes through the coverage region, interior nodes with an active sensor will change color, indicating that they now wait for the intruder. In the simulated scenarios, the intruder eventually exits the detection region. The simulation ends after the intruder exits. OMNeT++ writes node id, energy level, and other relevant statistics to an output file for analysis.

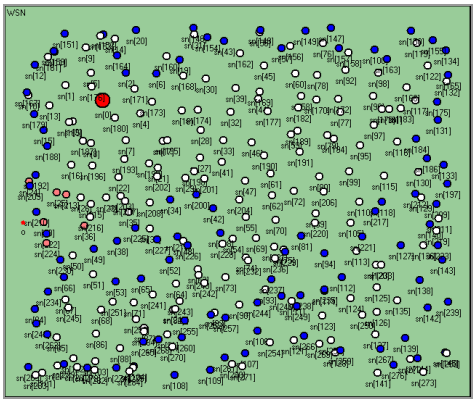

Figure 24: Tracking Phase

8.4 Metrics for Evaluation

Overhead analysis of a cryptographic protocol seeks to determine the protocol's footprint, which includes the increase in message size, the cost of the additional code on permanent and volatile storage, and the number of CPU cycles required to carry out the cryptographic processing. The increase in message size, called message expansion, results in longer activation of node radios as they transmit and receive longer packets. The computational complexity, measured in number of CPU cycles, results in increased utilization of node processors. Since OCO aims to increase network longevity by conserving energy, this simulation specifically evaluates the energy costs resulting from cryptographic processing and increases in message size. This simulation merges both metrics into one value, energy use per message byte. As with the original OCO simulation, the model also evaluates energy loss caused by activation of the node's sensor module when in detection mode or the node's radio module as it waits to receive a message. The simulation evaluates OCO and s(OCO) on the Mica2 mote platform, shown in Figure 25 [41]. While this platform has reached end of life, research provides a wealth of data on its performance. The TinySec study in [8] includes detailed analysis of the costs of TinySec-Auth on the Mica2 sensor node.

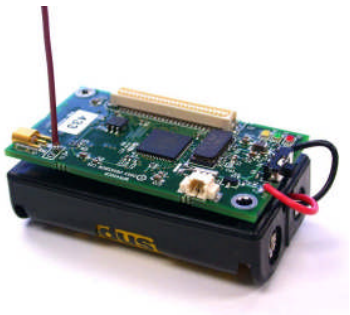

Figure 25: Mica2 Node [41]

The Mica2 mote draws its power from two AA batteries. As in [34], this simulation assumes use of high-energy alkaline manganese-dioxide batteries which start with a total rated capacity of 2850 milliamp hours (*mAh*). The batteries provide the node an average potential energy of 2.4 volts throughout node lifetime [42]. When the battery energy falls below 2.1 volts, nodes loose the ability to transmit or receive messages, effectively dying. This experiment simulates loading the Mica2 mote with two AA batteries by initializing the energy module with 24624 Joules. The initial energy calculation accounts for battery capacity and potential energy:

Joules = (2850 milliamp hours * 2.4 Volts * 3600 seconds) / 1000 = 24624.

Each simulation application includes a timer that periodically deducts energy consumed by the radio transceiver and the sensor board on active nodes. The simulated costs of operating these components are derived from analysis by Crossbow Technology, the manufacturer of the Mica mote series. Table 8 shows the operational cost of mote components as documented in the Mica2 datasheet [41]. The simulation only counts energy use by the radio in receive mode and by an active sensor board, since the energy use per byte metric incorporates processor drain and transmission power.

Table 8: Mica2 Energy Drain

Action	Energy (mJ/s)
Processor: Active Mode	19.2
Processor: Sleep Mode	0.036
Radio: Transmit at maximum power	64.8
Radio: Receive mode	24
Radio: Sleep mode	2.4
Sensor Board	1.68

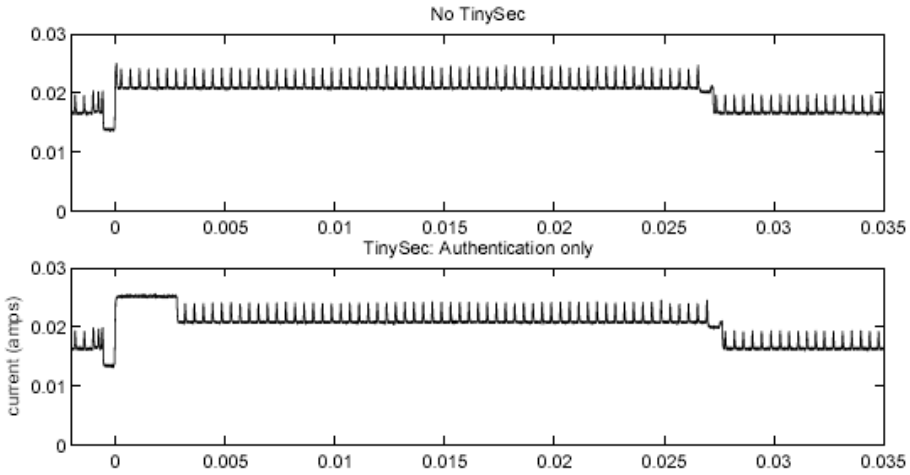

Figure 26: TinySec Power Drain [8]

Addition of cryptography into the communication scheme further increases energy consumption. The TinySec paper [8] documents the costs of the two main sources of energy drain during communication, message expansion overhead and cryptographic processing. Karlof calculated the costs of sending a 24-byte payload in TinyOS and TinySec-Auth to be 0.000160 *mAh* and 0.000165 *mAh* respectively. Their analysis, illustrated in Figure 26, shows an increase in power draw during the first segment of the authenticated transmission, since calculation of the MAC overlaps with sending the transmission start signal. Once the cryptographic operations complete, the power draw of a TinySec-Auth packet levels out to the same value as a TinyOS message. The TinySec-Auth packet continues to draw power a few milliseconds longer than the TinyOS packet as it transmits the additional byte resulting from the MAC. The Karlof analysis shows a three percent increase in energy consumption from the addition of authentication.

The simulation uses the 0.000160 *mAh* and 0.000165 *mAh* costs of sending a 24-byte payload to calculate the cost per byte of a TinyOS and a TinySec-Auth packet. These

values translate into costs of 43.20 μJ/byte and 44.55 μJ/byte for a TinyOS packet and a TinySec-Auth packet respectively, assuming that each byte requires the same energy draw. In actuality, shorter packets will bear more of the cost of computation, since the MAC is computed during the first microseconds of transmission. However, these actual costs cannot be easily determined without implementing the system in real hardware nodes. This simulation uses the assumed costs to calculate the cost per OCO and s(OCO) message.

Table 9: Message Cost

Message	OCO Length (B)	s(OCO) Length (B)	OCO Cost (µJ)	s(OCO) Cost (µJ)	Percent Increase
M1	7	7	302	302	0
M2	13	13	562	562	0
M3	15	16	648	713	10
M4	15	16	648	713	10
M5	13	14	562	624	11
M6	17	18	734	802	9
M7	13	14	562	624	11
M8	17	18	734	802	9
M9	11	12	475	535	13
M10	15	16	648	713	10
M11	11	12	475	535	13
M12	11	12	475	535	13
M13	17	18	734	802	9
M14	15	16	648	713	10

In the simulation, the application module deducts energy each time a node transmits or receives a message. The application deducts energy at rates outlined in Table 9. Note that the cost of authentication on an individual packet becomes higher as packet length decreases. The application also deducts energy when nodes maintain an active sensor module or radio module according to the schedule in Table 8. The simulation omits the TinySec-Auth costs of messages M1 and M2 because the risk assessment did not show sufficient justification for authentication messages in the Position Collection phase.

8.5 Scenario Setup

The integration of TinySec-Auth into OCO increases the cost of sending a packet. Although the TinySec paper identifies the cost of an individual message, it does not reveal how messaging affects the network as a whole. To estimate the costs of authentication in relation to other energy consuming services, the simulation runs scenarios with an intruder object that travels throughout the coverage region. These scenarios help contrast standard OCO with s(OCO) in relation to all energy consuming modules. The simulations distribute nodes over a 640 x 540-pixel region and contrast OCO with s(OCO) under multiple conditions. The changes in node count and intruder path make up the four experiments:

- Experiment 1: Distribute 500 nodes initially, track intruder through path 1 (see Figure 27).
- Experiment 2: Distribute 500 nodes initially, track intruder through path 2 (see Figure 28).
- Experiment 3: Distribute 1000 nodes initially, track intruder through path 1.
- Experiment 4: Distribute 1000 nodes initially, track intruder through path 2.

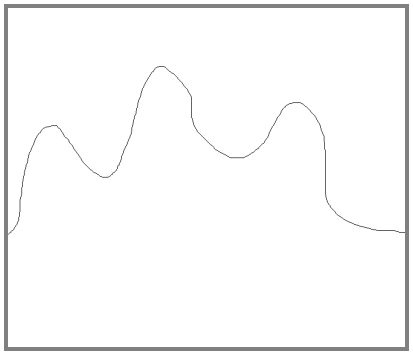

Figure 27: Intruder Path 1

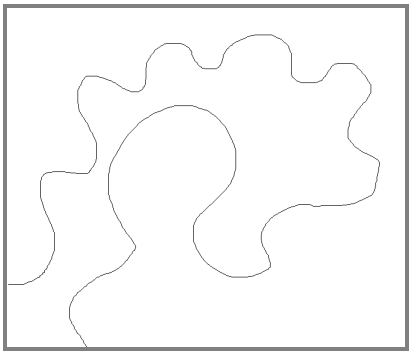

Figure 28: Intruder Path 2

Table 10: Simulation Scenarios

Nodes	500 Nodes	1000 Nodes
Path 1: OCO	Sample 1	Sample 5
Path 1: s(OCO)	Sample 2	Sample 6
Path 2: OCO	Sample 3	Sample 7
Path 2: s(OCO)	Sample 4	Sample 8

In each experiment, the simulation measures the cost of messaging during the Position Collection, Processing, and Tracking phases. One sample in each experiment simulates the cost of using only unauthenticated OCO. The second sample simulates s(OCO) and authenticates all Processing and Tracking phase messages. Table 10 summarizes the eight simulation scenarios.

The Tracking phase separately models the paths of two intruders. In the first simulation, an intruder enters the detection region from the left and follows the path illustrated in Figure 27. This simulated intrusion lasts for five minutes. In the second simulation, the intruder circles throughout the grid for 12 minutes as depicted in Figure 28. During the simulation, interior nodes will activate their sensor module upon receipt of an alert message from a border node. Their sensor remains active until the intruder moves out of their sensing radius.

Once the intruder exits the coverage grid, the simulation terminates. The Tracking application records the path of the intruder, the id of nodes that sensed the intruder object, and timestamp for each sensing event. The simulation also saves the node id and total remaining energy for each node. This simulation enables assessment of the energy consumption characteristic of OCO and s(OCO). The experiment evaluates results from

each of the eight simulation runs with a focus on total energy consumed per node, per run, and per OCO phase. This allows determination of the cost of security in relation to other sensor node components.

9 Experimental Results

This analysis of results aims to identify the impact s(OCO) has on individual nodes and on the network as a whole. According to the TinySec paper, the addition of authentication increases the cost of sending a single 24-byte packet by three percent. However, this value does not apply globally to individual OCO or s(OCO) packets, which range in length from between 12 and 18 bytes total. Shorter packets cost more because MAC computation must occur early in packet transmission, before the first byte leaves the mote radio. In longer packets, the cost of computing the MAC averages out over more bytes. This supports the case for defining a lower bound of three percent. Other services besides messaging conume energy in a wireless sensor network, such as the sensor and processor. The experimental results support the hypothesis that s(OCO) costs between three percent and thirteen percent more of total network energy than the standard, unauthenticated OCO.

The 500 node experiments and the 1000 node experiments originate from wholly independent populations, thus, their results should not be compared when evaluating energy consumption. However, their results reveal an interesting trait of OCO. Both the 500-node and the 1000-node networks cover the same size of detection region, represented in the simulation by a 640x540-pixel grid. The 1000-node network requires approximately the same number of activated motes as the 500-node network, yet it produces a smoother

border. The 1000-node network actually consumes less energy during the simulation than the 500-node network. Also note that the comparisons between OCO and s(OCO) start with output from the 500 node and 1000 node Position Collection simulation. This provides assurance that node id, position, and occupation remain consistent throughout the experiments.

9.1 Energy Consumption per Node

The first phase of the analysis focuses on the amount of energy consumed by each node during the simulation. This allows identification of conditions that exacerbate node energy consumption. s(OCO) should not cause nodes to prematurely run out of power. Figure 29, showing the energy consumed by each active node, illustrates how closely s(OCO) mirrors OCO. The subsequent histograms show the distribution of energy consumption of all member nodes, including forwarding nodes, border nodes, and redundant nodes.

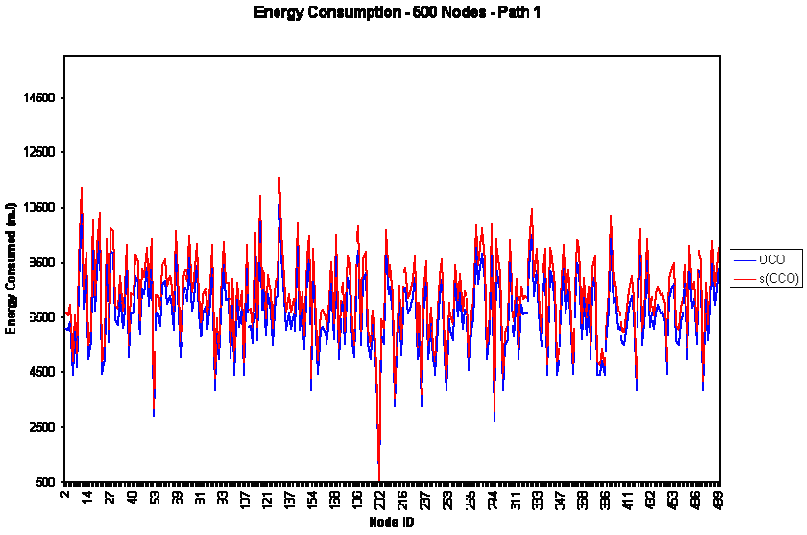

Figure 29: Energy Consumption per Node - Line Graph

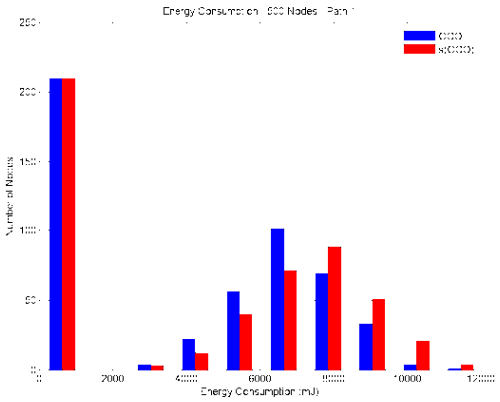

Figure 30: Energy Consumption – All Nodes - Experiment 1 (500 Nodes, Path 1)

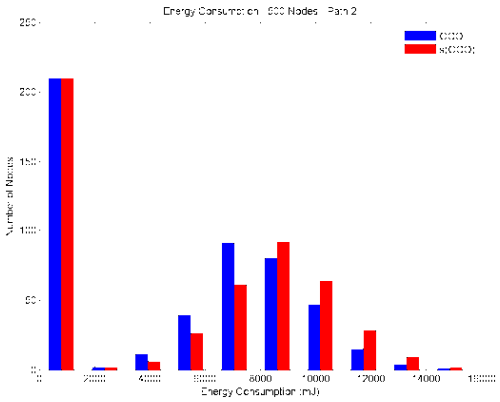

Figure 31: Energy Consumption – All Nodes - Experiment 2 (500 Nodes, Path 2)

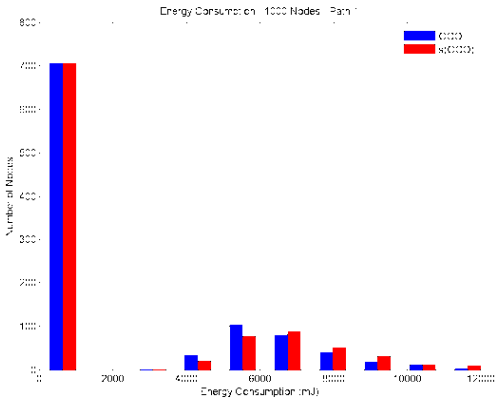

Figure 32: Energy Consumption – All Nodes - Experiment 3 (1,000 Nodes, Path 1)

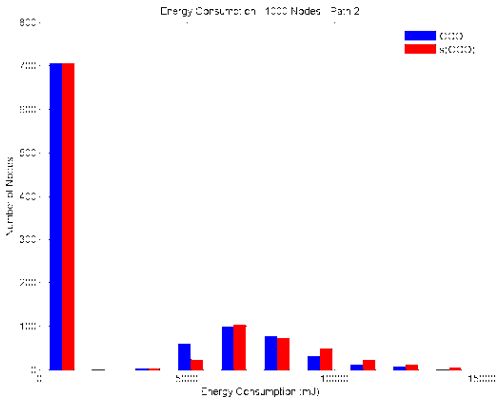

Figure 33: Energy Consumption – All Nodes - Experiment 4 (1,000 Nodes, Path 2)

Figure 30 and Figure 31 show a comparison between energy consumption in OCO and energy consumption per node in s(OCO) for experiments 1 and 2, the 500 node experiments. Figure 32 and Figure 33 show a comparison between energy consumption in OCO and energy consumption in s(OCO) for experiments 3 and 4, the 1000 node experiments. The most prominent feature of the graphs, the cluster of nodes that consume less than 1000 mJ, represents the nodes assigned a redundant node occupation. This large quantity of nodes selected as border nodes skews the mean energy consumption for the

remaining nodes in the experiments, but it also highlights the ability of OCO to reserve nodes until needed. In the 500-node experiment depicted in Figure 30, the mean energy consumption for the OCO nodes is 3996 mJ, well below the average energy consumption by nodes that participate in the Processing and Tracking phases.

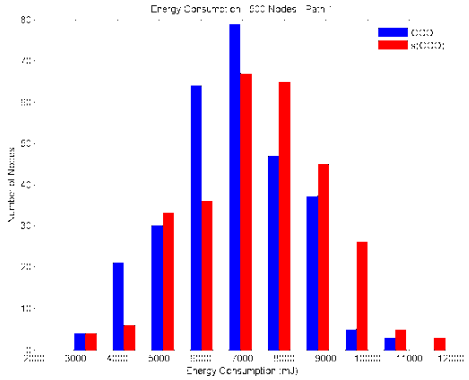

Figure 34: Energy Consumption – Active Nodes – Experiment 1 (500 Nodes, Path 1)

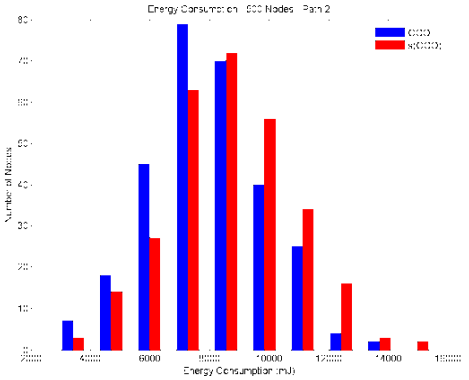

Figure 35: Energy Consumption – Active Nodes – Experiment 2 (500 Nodes, Path 2)

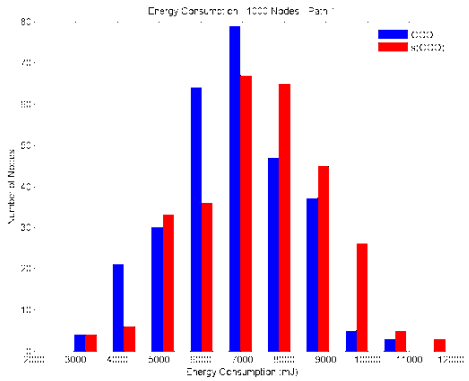

Figure 36: Energy Consumption – Active Nodes – Experiment 3 (1,000 Nodes, Path 1)

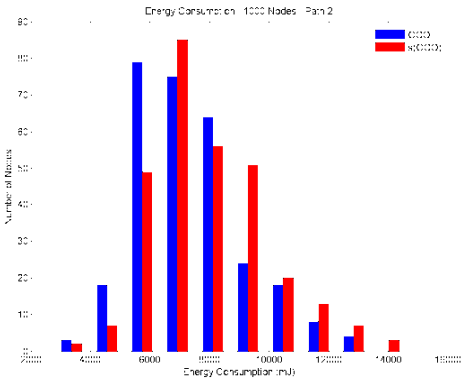

Figure 37: Energy Consumption – Active Nodes – Experiment 4 (1,000 Nodes, Path 2)

Figure 34, Figure 35, Figure 36, and Figure 37 offer a different perspective on energy consumption by omitting redundant nodes from the calculation. They depict the same experiments as shown in Figure 30 through Figure 33, but leave redundant nodes out of the calculation. The experiments justify this omission because the redundant nodes never

play a role in simulations that require authentication. The graphical results make more sense since the mean energy consumption for the remaining nodes matches the plotted histogram. The short experiments consume no more than 0.0352% of a node's energy on average. Table 11 shows the mean energy consumption per node for nodes that remain through the Processing and Tracking phases.

Table 11: Mean Energy Consumption per Node (mJ)

Nodes	500 Nodes	1000 Nodes
Path 1: OCO	6779	6480
Path 1: s(OCO)	7406	7096
Path 2: OCO	7971	7320
Path 2: s(OCO)	8667	7987

The mean energy consumption per node for nodes used in the Processing and Tracking phases matches the hypothesized values. In the 1000-node simulation, experiment 3, tracking an intruder across path 1 requires 9.51% more energy with message authentication than without. This is the maximum observed in these experiments. The lowest observed difference in mean energy consumption, 8.73%, occurs in experiment 2 (500 nodes, path 2). The results indicate that the longer a simulation runs, the less significant the difference between OCO and s(OCO). Longer simulations that run for hundreds of simulation hours should show the difference between OCO and s(OCO) approaching the three percent bound.

Table 12: Tests for adequate pairing

Nodes	r	T-Test	DoF
Experiment 1 (500 Nodes, Path 1)	0.9997	2.4378E-190	289
Experiment 2 (500 Nodes, Path 2)	0.9989	5.2301E-171	289
Experiment 3 (1,000 Nodes, Path 1)	0.9997	4.7574E-174	292
Experiment 4 (1,000 Nodes, Path 2)	0.9990	5.7405E-173	292

Table 12 strengthens the results by showing that the OCO and s(OCO) test groups originate from the same population. In each experiment, the comparison between OCO and s(OCO) uses the same output from that experiment's Position Collection application. The Pearson correlation coefficient (r) shows a minimal degree of variance between the OCO test group and the s(OCO) test group. The t-test value shows a very low chance that the results from the test would occur by chance. The degrees of freedom (DoF) indicates the number of nodes being tested minus 1.

9.2 System Energy Consumption

The remaining analysis builds a case showing the total cost of integrating authentication into an OCO network. It uses messaging metrics tracked during the simulation to break down the total amount of energy consumed into the energy drawn by OCO messaging, by authentication, and by other components. This approach provides two benefits to the analysis of OCO. First, it provides a baseline to help derive the cost of authentication in comparison to non-communication operations. Second, it highlights characteristics of OCO communication patterns throughout different phases of network organization and tracking. While the communication patterns do not play a direct role in the OCO/s(OCO) comparison, they do highlight communications that would benefit from an alternative, non-cryptographic, risk mitigation strategy. Table 13 summarizes the total energy consumed by the network during each simulation, aggregating the cost of

messaging, sensor activation, and processor usage. When viewed in terms of the total amount of energy allocated to each simulation, these short experiments consume very little. The 500-node experiment starts with 12,312,000 Joules available for the simulation. The 1000-node simulation begins with 24,624,000 Joules.

Table 13: Total Energy Consumed

Nodes	500 Nodes (J)	500 Nodes (%)	1000 Nodes (J)	1000 Nodes (%)
Path 1: OCO	1,933	0.0157	1,900	0.0077
Path 1: s(OCO)	2,115	0.0172	2,080	0.0084
Path 2: OCO	2,279	0.0185	2,146	0.0087
Path 2: s(OCO)	2,481	0.0202	2,341	0.0095

The 1000 node simulation consumes less energy overall because 1000 nodes yield a smoother, more well defined border than 500 nodes. Because of the porous perimeter, the 500-node simulation has nodes fulfilling the border node occupation that actually reside within the interior of the detection region. These nodes continually draw power throughout the experiment. The 1000 node simulation requires only 60 fully activated border nodes, whereas the 500-node simulation requires 98 nodes to fulfill the border node occupation.

In order to break down the total energy cost from Table 13 into components, the simulation code tracks the number of messages sent and received by each node in the network. The analysis uses the count of messages to derive the amount of energy spent on messaging. Each experiment that compares OCO with s(OCO) exchanges the same number of messages with the same content. The only difference is the cost metric assigned to each message. The system quickly exchanges a large number of messages during network organization. The Position Collection and Processing phases take less than two seconds in

simulation time, yet they account for more than 95% of messages sent and received. Once the network topology stabilizes, OCO and s(OCO) can operate efficiently for a long period of time. Table 14 and Table 15 show the total number of each message sent and received during the simulation.

Table 14: Messages Sent in Path 1 Simulation

	500 Nodes		1000 Nodes	
Message	**Sent**	**Received**	**Sent**	**Received**
M1	499	2,542	999	4,839
M2	9,155	51,932	36,949	194,986
M3	84,681	84,694	85,849	85,859
M4	28,420	12,180	17,520	9,636
M5	43,790	815,977	58,400	809,862
M6	815,660	273,935	809,722	165,334
M7	117,327	421,924	90,913	550,861
M8	11,142	1,290	11,576	1,268
M9	111,240	12,620	101,228	10,300

Table 15: Messages Sent in Path 2 Simulation

	500 Nodes		1000 Nodes	
Message	**Sent**	**Received**	**Sent**	**Received**
M1	499	2,542	999	4,839
M2	9,155	51,932	36,949	194,986
M3	84,681	84,694	85,849	85,859
M4	28,420	12,180	17,520	9,636
M5	43,790	815,977	58,400	809,862
M6	815,660	273,935	809,722	165,334
M7	117,327	421,924	90,913	550,861
M8	35,289	3,345	31,978	2,949
M9	360,223	33,841	280,780	27,836

The analysis attempts to build the cost of simulation from the bottom up. It multiplies the number of each message sent and received by each node with the cost of that message. The product is the value for total message cost. Since nodes do not change occupations during a these simulations, the cost of sensor, processor, and radio usage can be determined by multiplying the simulation time by the cost of operating the device per second. Summing the messaging costs with the device operation costs yields a total network operation cost. These results provide assurance that the total cost calculated by hand matches the total cost reported in the simulation.

The product of the count of messages sent and received times the cost per message gives the cost of standard OCO messaging without authentication. The cost of operating radio and sensor boards can be derived from the cost of operating the respective devices per

second and the simulation time. Table 16 shows the basic cost of OCO communication in the Messaging column. The sensor and radio operational costs appear next. The authentication column represents the increase in energy use incurred by adding TinySec-Auth on top of OCO messaging. The last column represents this cost increase as a percentage.

Table 16: Total Energy Use

	Messaging (J)	Sensor & Radio (J)	Authentication (J)	Total (J)	Cost Increase
500 Nodes					
Path 1	1,825	108	182	2,115	9.42%
Path 2	1,973	306	202	2,481	8.86%
1,000 Nodes					
Path 1	1,888	12	180	2,080	9.47%
Path 2	1,998	148	195	2,341	9.09%

As shown above, the addition of an authentication only solution clearly exceeds the benchmark identified for sending a 24-byte payload as indicated in the TinySec paper. The total energy consumed by the network lies within the three percent lower bound and thirteen percent upper bound suggested in the hypothesis.

9.2.1 Patterns of energy use

Each phase of OCO / s(OCO) network organization and communication possesses unique communication patterns. The following contour diagrams illustrate the cost of each phase of communications in terms of a node's location in the network. In the following figures, the base station is located at x, y coordinate (403,140). During the Position Collection phase, nodes located near the base station consume significantly more energy as they pass messages between the base station and exterior nodes, as shown in Figure 38.

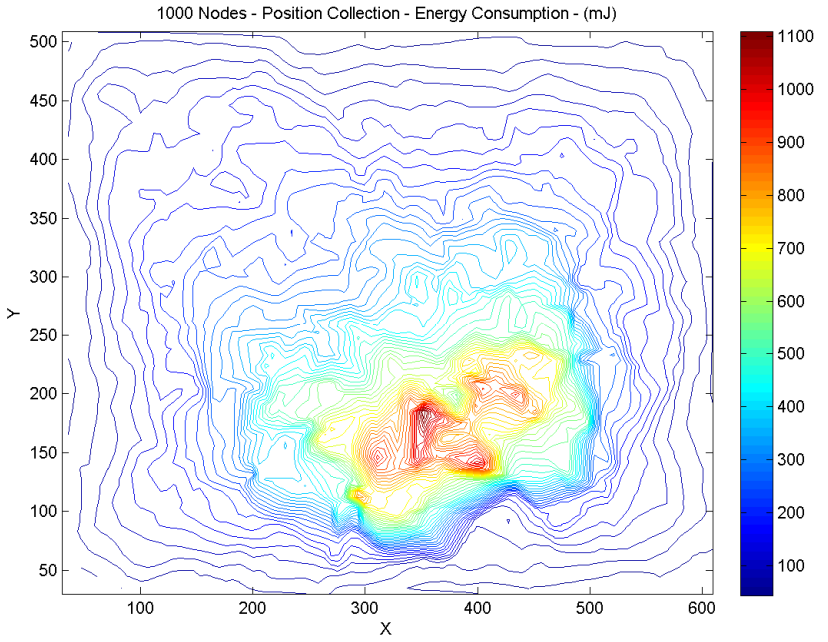

Figure 38: Contour diagram - 1000 Nodes - Position Collection

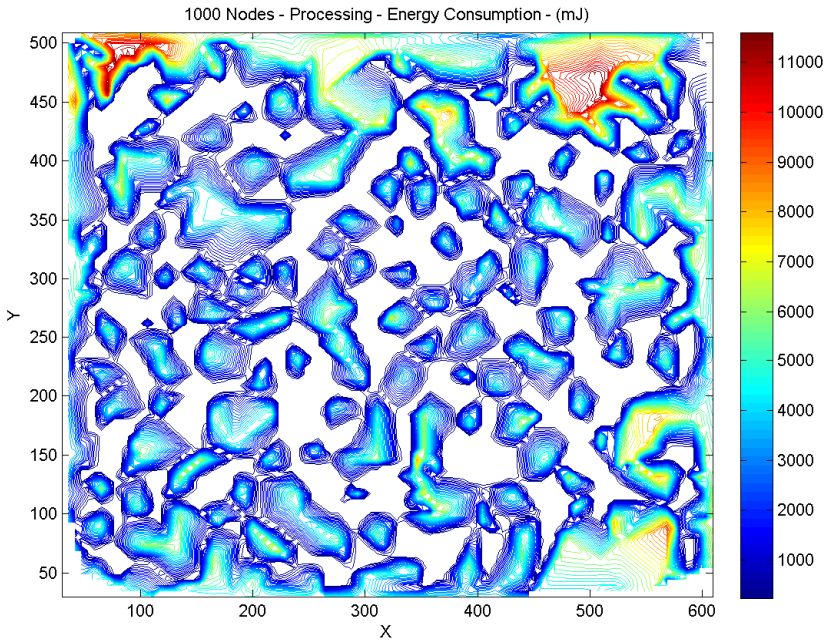

Figure 39: Contour diagram - 1000 Nodes – Processing

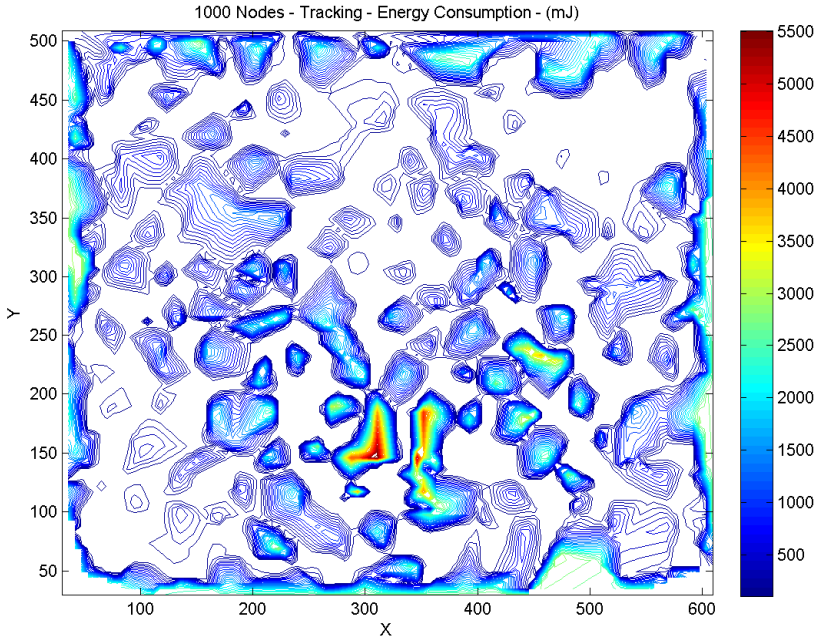

Figure 40: Contour diagram - 1000 Nodes - Tracking

Figure 39 shows the cost of messaging spread evenly throughout the network during the Processing phase. Despite the large number of messages broadcast by the base station, the energy consumption remains evenly distributed since nodes only forward assignments toward the perimeter and not send results back to the base. As shown in Figure 40, the Tracking phase consumes significantly less energy than the network organization phases. It draws the most power near the base station and along the perimeter. While these contour diagrams do not differentiate between OCO and s(OCO), they do add value in the s(OCO) risk mitigation strategy. Nodes that transmit an unusually large number of messages are especially sensitive to the increased cost of authentication.

9.3 Discussion of Results

The addition of authentication to OCO increases total energy consumption to between 8 and 10 percent of all energy consumed during the experiments. While the results

only slightly exceed the costs of maintaining an active processor or an active sensor, they still negatively influence the longevity of an OCO network. Since the network stabilizes during the Tracking phase, it should be able to sustain operations for a long duration with s(OCO). Nodes that transmit or receive a higher number of messages than its peers may benefit from an alternative risk assessment methodology. For example, nodes located near the base station will receive more intrusion alert messages from the perimeter than other nodes. Employing filtering of redundant alerts in forwarding nodes near the perimeter will reduce the burden on forwarding nodes near the base. In another example, the risk assessment assigned a high risk to all messages in the Processing phase, primarily because of the large attack window. Nodes accept messages in the Processing phase through most of the network lifetime. If the total risk value assigned to each message could be reduced, the network could survive without authentication during the Processing phase. One way to achieve this is to narrow the attack window. The base station could send authenticated messages that define the beginning and conclusion of the Processing phase.

10 Conclusion

The survey conducted as part of this thesis tracked the evolution of two disparate fields in sensor network research: target tracking applications and authentication protocols. While both areas of research strive for efficiency, the addition of security to a target tracking mechanisms increases energy consumption. This increase contradicts the quest of target tracking research to uncover maximum efficiency. Target tracking methods such as GDAT and OCO improve network longevity by forcing the majority of nodes to sleep. These techniques demonstrate superiority in efficiency while maintaining a high level of accuracy in tracking intruders. Sensor network authentication protocols similarly strive for efficiency by reducing computation and communication costs. Classic proposals such as TinySec and SPINS integrate easily with any target tracking method. However, the application designer must carefully evaluate the security requirements of an application. Blanket coverage of all communications with a security countermeasure unnecessarily reduces the ability of efficient target tracking mechanisms to provide long-lasting networks.

This thesis demonstrated the cost of a very simple authentication mechanism. This mechanism leaves vulnerabilities exposed. For example, Zhu, et al. [40], demonstrate that simple peer-to-peer authentication protocols like TinySec-Auth suffers fatal flaws in the presence of node compromise. An attacker can overtake an active node, gain access to keying material, and falsely inject messages into the network. Providing protection for this

level of attack will require even more energy consuming components, such as secure key rotation and interleaved authentication.

10.1 Future Research

Image processing techniques clearly provide superior perimeter intrusion detection, yet threats against target tracking methods can reduce their trustworthiness. The security mechanisms in this thesis provide countermeasures that reduce the risk of these threats. However, the research also opens avenues for future research. The three primary opportunities include efficiently providing interior nodes with active sensors, and adding additional security countermeasures to s(OCO).

The GDAT proposal demonstrated an efficient mechanism to provide interior intrusion detection at the cost of having a porous perimeter. OCO can efficiently track intruders through the perimeter as long as they enter by crossing the border. However, an intruder can enter the detection region from within the border. In a combat scenario, troops or military equipment can be airlifted into the combat zone deep within the detection radius of border nodes. While GDAT provides a mechanism for this by rotating guard duty among nodes in a cluster, OCO does not posses a clustered topology that supports easy integration of the two methods. One could activate the sensor on forwarding nodes, but this would cause forwarding nodes to prematurely run out of power and result in unnecessary network reorganization. As an alternative, the OCO protocol could inform forwarding nodes of nearby redundant nodes that it could temporarily activate to provide interior coverage.

The security mechanism simulated in this thesis provides a reasonable amount of security in consideration of the threats outlined in the risk assessment. However, there remains a significant threat if an attacker can compromise legitimate nodes. An attacker can reverse engineer a compromised node to uncover cryptographic keying material. Since the TinySec evaluation and this simulation use a single network shared key, the threat of node compromise ranks high. Fortunately, TinySec provides a modular design that simplifies integration with various key distribution mechanisms. Like the authentication mechanisms, the keying mechanism should be tailored to the characteristics of the application. Since OCO keeps many nodes in a sleeping state until needed, rekeying could cause redundant nodes to have to wake to receive new keys, thus unnecessarily depleting energy. Future research should identify a re-keying mechanism appropriate for OCO.

If nodes are compromised while still active, the attacker can maintain access that allows them to capture newly broadcast keys. The Zhu, et al., proposal offers countermeasures that protect cluster-based networks against data injection by compromised nodes. This requires clusters to contain a minimum number of nodes. OCO does not possess a clustered topology, and thus does not provide a simple fit for Zhu's interleaved authentication. Future research can identify mechanisms that allow multiple nodes to collaborate to determine whether an intrusion has occurred.

REFERENCES

[1] Tran, S. P. and Yang, T. A., "OCO: Optimized Communication & Organization for Target Tracking in Wireless Sensor Networks," *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing,* vol. 1, pp. 428-435, 2006.

[2] Yang, T. A. and Nguyen, T. A., "Network security development process: a framework for teaching network security courses," *Journal of Computing Sciences in Colleges,* vol. 21, pp. 203-209, 2006.

[3] Perrig, A., Szewczyk, R., Tygar, J. D., Wen, V., and Culler, D. E., "SPINS: Security Protocols for Sensor Networks," *Wireless Networks,* vol. 8, pp. 521-534, 2002.

[4] Varga, A., "OMNeT++ Discrete event simulation system. User Manual," *Technical University of Budapest, Dept. of Telecommunications,* 2006.

[5] Kung, H. T. and Vlah, D., "Efficient location tracking using sensor networks," *IEEE Wireless Communications and Networking, WCNC,* vol. 3, pp. 1954-1961 vol.3, 2003.

[6] Heinzelman, W. R., Chandrakasan, A., and Balakrishnan, H., "Energy-efficient communication protocol for wireless microsensor networks," *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences,* p. 10, 2000.

[7] Rababaah, H. and Shirkhodaie, A., "Guard Duty Alarming Technique (GDAT): A Novel Scheduling Approach for Target-tracking in Large-scale Distributed Sensor Networks," *IEEE International Conference on System of Systems Engineering,* pp. 1-6, 2007.

[8] Karlof, C., Sastry, N., and Wagner, D., "TinySec: a link layer security architecture for wireless sensor networks," *Proceedings of the 2nd international conference on embedded networked sensor systems,* pp. 162-175, 2004.

[9] Pister, K., "29 Palms fixed/mobile experiment: Tracking vehicles with a UAV delivered sensor network," 2001.

[10] Romer, K. and Mattern, F., "The design space of wireless sensor networks," *IEEE Wireless Communications,* vol. 11, pp. 54-61, 2004.

[11] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K., "System architecture directions for networked sensors," *ACM SIGPLAN Notices,* vol. 35, pp. 93-104, 2000.

[12] Tran, S. P., Yang, T. A., Cao, D., and Nguyen, T. A., "OCO: An efficient method for tracking objects in wireless sensor networks," 2006.

[13] Bishop, M., *Computer security: art and science*. Boston, MA: Addison-Wesley, 2003.

[14] Schneier, B., *Applied cryptography : protocols, algorithms, and source code in C,* 2nd ed. New York: Wiley, 1996.

[15] Wood, A. D. and Stankovic, J. A., "Denial of service in sensor networks," *IEEE Computer,* vol. 35, pp. 54-62, 2002.

[16] Wood, A. D., Fang, L., Stankovic, J. A., and He, T., "SIGF: a family of configurable, secure routing protocols for wireless sensor networks," *Proceedings*

of the fourth ACM workshop on Security of ad hoc and sensor networks, pp. 35-48, 2006.

[17] Luk, M., Perrig, A., and Whillock, B., "Seven cardinal properties of sensor network broadcast authentication," *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, pp. 147-156, 2006.

[18] Harris, S., *Mike Meyers' CISSP Certification Passport with CDROM*: Osborne/McGraw-Hill, 2002.

[19] Wang, X. and Yu, H., "How to Break MD5 and Other Hash Functions," *Advances in Cryptology – EUROCRYPT 2005*, pp. 19-35, 2005.

[20] Rivest, R., "The MD5 Message-Digest Algorithm, RFC 1321," *IETF*, 1992.

[21] Eastlake, D. and Jones, P., "US Secure Hash Algorithm 1 (SHA1), RFC 3174," *IETF*, 2001.

[22] Wang, X., Yin, Y. L., and Yu, H., "Collision search attacks on SHA1," *Crypto 2004*, August 15-19, 2004.

[23] Bellare, M., Kilian, J., and Rogaway, P., "The security of the cipher block chaining message authentication code," *Journal of Computer and System Sciences*, vol. 61, pp. 362-399, 2000.

[24] NIST, "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication," 2005.

[25] Michail, H. E., Kakarountas, A. P., Selimis, G., and Goutis, C. E., "Throughput optimization of the Cipher Message Authentication Code," *15th International Conference on Digital Signal Processing*, pp. 495-498, 2007.

[26] Rivest, R. and Baldwin, R., *The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms, RFC2040*: RFC Editor, 1996.

[27] NIST, "Skipjack and KEA algorithm specifications," 1998.

[28] "FIPS 197, Advanced encryption standard (AES)," 2001.

[29] Roman, R., Alcaraz, C., and Lopez, J., "A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes," *Mobile Networks and Applications*, vol. 12, pp. 231-244, 2007.

[30] Grossschadl, J., Tillich, S., Rechberger, C., Hofmann, M. A. H. M., and Medwed, M. A. M. M., "Energy Evaluation of Software Implementations of Block Ciphers under Memory Constraints," in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07*, 2007, pp. 1-6.

[31] Law, Y. W., Doumen, J., and Hartel, P., "Survey and benchmark of block ciphers for wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, pp. 65-93, 2006.

[32] Choi, K. J. and Song, J.-I., "Investigation of feasible cryptographic algorithms for wireless sensor networks," *The 8th International Conference on Advanced Communication Technology, ICACT*, vol. 2, p. 3 pp., 2006.

[33] Daemen, J. and Rijmen, V., "The Block Cipher Rijndael," *Proceedings of the International Conference on Smart Card Research and Applications*, 2000.

[34] Luk, M., Mezzour, G., Perrig, A., and Gilgor, V., "MiniSec: a secure sensor network communication architecture," *Proceedings of the 6th international conference on Information processing in sensor networks*, pp. 479-488, April 25 - 27, 2007.

[35] Alliance, Z., "Zigbee specification. Technical Report Document 053474r06," 2005.

[36] Rogaway, P., Bellare, M., Black, J., and Krovetz, T., "OCB: a block-cipher mode of operation for efficient authenticated encryption," *Proceedings of the 8th ACM conference on Computer and Communications Security*, 2001.

[37] Wood, A. D. and Stankovic, J. A., "AMSecure: secure link-layer communication in TinyOS for IEEE 802.15.4-based wireless sensor networks," *Proceedings of the 4th international conference on Embedded networked sensor systems*, 2006.

[38] "IEEE 802.15.4-2003. Wireless MAC and PHY Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)," 2003.

[39] Qi, X. and Ganz, A., "Runtime security composition for sensor networks (SecureSense)," *IEEE 58th Vehicular Technology Conference, VTC*, vol. 5, pp. 2976-2980 Vol.5, 2003.

[40] Zhu, S., Setia, S., Jajodia, S., and Peng Ning, A., "An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks," *IEEE Symposium on Security and Privacy*, pp. 259-271, 2004.

[41] Inc., C. T., "MICA2 Datasheet."

[42] Landsiedel, O., Wehrle, K., and Gotz, S., "Accurate prediction of power consumption in sensor networks," *The Second IEEE Workshop on Embedded Networked Sensors, EmNetS-II*, pp. 37-44, 2005.