Graduate School ETD Form 9
(Revised 01/14)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Saharnaz Zare Afifi

Entitled
SECURING SENSOR NETWORK

For the degree of MASTER OF SCIENCE IN ELECTRICAL & COMPUTER ENGINEERING

Is approved by the final examining committee:

Dr. Brian King

Dr. Maher Rizkalla

Dr. Paul Salama

To the best of my knowledge and as understood by the student in the *Thesis/Dissertation Agreement. Publication Delay, and Certification/Disclaimer (Graduate School Form 32)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Dr. Brian King

Approved by Major Professor(s): _____

_____

Approved by: Dr. Brian King 06/30/2014

Head of the Department Graduate Program Date

SECURING SENSOR NETWORK

A Thesis

Submitted to the Faculty

of

Purdue University

by

Saharnaz Zare Afifi

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

August 2014

Purdue University

Indianapolis, Indiana

I would like to dedicate this work to my father, who motivated and guided me in my youth to pursue an engineering career; I know he would be so proud of me if he were able to see what I have accomplished.

ACKNOWLEDGMENTS

I would like extend my deepest gratitude towards Dr. Brian King for all of his continuous support from the beginning through the completion of this research. Without his guidance and patience, this research would not have been possible.

I would also like to thank Dr. Paul Salama and Dr. Maher Rizkalla for accepting my invitation to be committee members. I appreciate their constructive feedback and suggestions for improving my thesis. In addition, I am thankful for Mrs. Sherrie Tucker, her knowledge and unparalleled kindness gave me the support I desperately needed to complete my work. I would also like to recognize the entire Electrical and Computer Engineering department for their assistance, as well as providing the necessary facilities and equipment for all graduate work.

I am thankful for all the support from my family while they aided me emotionally and financially for this stage of my life.

This work was supported by a grant, award number N00164-11-1-2005, from the Office of Naval Research and NSWC Crane. I would also like to acknowledge Indiana University, for their financial support.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Zare Afifi, Saharnaz MSECE, Purdue University, August 2014. Securing Sensor Network. Major Professor: Brian King.

A wireless sensor network consists of lightweight nodes with a limited power source. They can be used in a variety of environments, especially in environments for which it is impossible to utilize a wired network. They are easy/fast to deploy. Nodes collect data and send it to a processing center (base station) to be analyzed, in order to detect an event and/or determine information/characteristics of the environment.

The challenges for securing a sensor network are numerous. Nodes in this network have a limited amount of power, therefore they could be faulty because of a lack of battery power and broadcast faulty information to the network. Moreover, nodes in this network could be prone to different attacks from an adversary who tries to eavesdrop, modify or repeat the data which is collected by other nodes. Nodes may be mobile. There is no possibility of having a fixed infrastructure. Because of the importance of extracting information from the data collected by the sensors in the network there needs to be some level of security to provide trustworthy information.

The goal of this thesis is to organize part of the network in an energy efficient manner in order to produce a suitable amount of integrity/security. By making nodes monitor each other in small organized clusters we increase security with a minimal energy cost. To increase the security of the network we use cryptographic techniques such as: public/ private key, manufacturer signature, cluster signature, etc. In addition, nodes monitor each other's activity in the network, we call it a "neighborhood watch". In this case, if a node does not forward data, or modifies it, other nodes which are in their transmission range can send a claim against that node.

1. INTRODUCTION

Sensors can play an important role in one's life. The simplest example is a thermostat so we can have the optimal indoor temperature while saving energy. Sensors, like oxygen sensors in automobiles, can help us provide a cleaner environment, by monitoring the environment and providing timely, accurate feedback. Sensors make our lives easier and more comfortable, however, faulty sensors can be life threatening. In 2011, a plane crashed (Cork plane crash) because of a faulty sensor in the airplane engine; the sensor provided wrong information about the pressure and temperature to the fuel control unit. Investigators found that the sensor was showing a temperature value up to $57°C/135°F$ below the actual performance of the engine, which caused the aircraft to: roll rapidly left and right, turn upside-down, and miss the runway completely during landing. The end result of the faulty sensor was 6 dead and multiple serious injuries [1].

Consider a setting where a city has a number of heterogeneous sensors located throughout the city, and they are used to monitor the environment to protect it, see Fig.1.1. For example, there may be sensors in power grid area, industrial monitoring sensors, underground water sensors to check the quality of underground water, sensors in subway transportation centers, used for measuring the quality of the air or smoke. Suppose all of these sensors are connected to some emergency dispatch center. Now, the emergency dispatch center completely relies on the sensors' information in order to determine an emergency situation. The center receives data from the sensors and determines any emergency status. Based on the information that sensors provide to the center, the center will determine the emergency and the location for emergency responders to respond to. If for instance a fire hazard is occurring in a subway station and the sensors are faulty, the correct warning and/or location may not be determined by the emergency dispatch center and which can cause a significant problem. In Fig.

1.1, we illustrate a wireless sensor monitor environment and its surroundings. In case of hazardous behavior, the center informs the Emergency Dispatch Center. Some examples of sensors: gas sensor (in a Subway/Transportation center), Motion detector (Surveillance Building, Power Grid), sensors can determine CL, pH, Hg, CO_2, O_2 of water (Water Treatment Center, Lake).

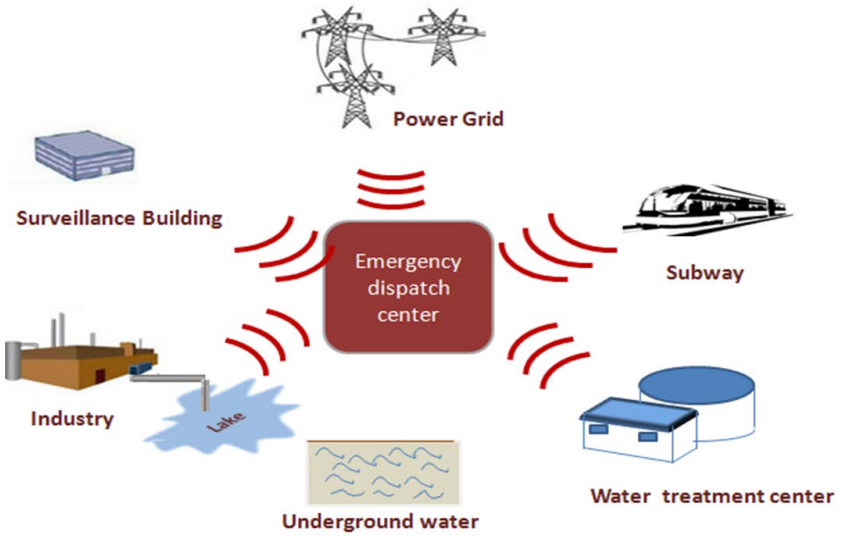

Fig. 1.1.: A wireless sensor monitor environment.

In our work, we assume we have a field of heterogeneous mobile sensors with no trusted third party/no central authority which form a mobile ad-hoc network. The lack of infrastructure is what makes this network different from a "traditional network". If the field is large enough, not all the nodes are within each others' transmission range. The nodes will use a cooperative communication protocol to transmit data to the base station which is typically in an ad-hoc network. Because nodes are mobile, the topology of the network is not consistent and will change over time. The functionality of the network should be divided or distributed between nodes to reduce the vulnerability of the network. Without additional security mechanisms, the mobile ad-hoc network is not secure because nodes can become compromised/faulty,

etc. Furthermore, there would be a number of problems concerning trust of information transmitted. An adversary can easily access information and/or data inside the network [2]. For security concerns and avoiding data leakage, nodes need to utilize encryption and authentication mechanisms in order to communicate in a secure manner. Due to sensors' lack of a renewable energy source, we need to construct mechanisms in an energy efficient manner.

In our work, sensors will organize the aspect of the network autonomously. From this "organization'', we construct secure energy efficient mechanisms.

2. BACKGROUND

In this chapter, two common topologies of a sensor network will be discussed. We will discuss their pros and cons and what topology is proper for our sensor network. Furthermore, we will discuss different types of attacks on the network. Lastly, we discuss many of the cryptographic tools used in our work.

2.1 Different Topologies for A Network

There are many different topologies for a sensor network. Based on the network performance and network usage we selected a proper topology to make our network more efficient. Network topologies are typically based on following [3]:

1. **Energy Efficiency:** The functionality of a sensor network should be extended as long as possible. The network topology has an important effect on energy usage and the way the sensors communicate with each other affects the efficiency of the network. If we can minimize the energy usage of each sensor we would save energy for the whole network.

2. **Network Lifetime:** A network's lifetime is calculated as when the first node dies (its energy has drained). Many of the network topologies are based on developing a topology to use the same amount of energy in all the nodes in a network.

3. **Data Accuracy:** The notion of data accuracy depends on the network application.

4. **Latency:** Latency is defined as the delay in data transmission routing and data aggregation [3]. It can be measured by the amount of time between data generation in source nodes and data receiving in the base station.

2.1.1 Flat Network

A flat network is a network for which all of the nodes in the network are connected to the base station. In a flat network, each node plays the same role as other nodes in the network [3]. Also, all the nodes have the same battery usage. In this kind of network, data aggregation would be in a way that sink will send the query message to the sensors in the network via flooding and sensors which have data matching the query, send a respond back to the sink.

Some disadvantages of a flat network are: it requires significant amount of communication and computation to the sink [3], if the sink is not connected to a renewable source of energy the battery will be dead and it will cause the death of the whole network. Also the communications between nodes would flood in the whole network, which is again not an energy efficient procedure. Another drawback of a flat network is inability to guaranty the data delivery. In addition, topology is suitable for a small network, where all the nodes are at most one hop away from the sink. As you can see in the Fig. 2.1, all nodes are one hop away from the sink, circles in the figure represent sensor/ nodes. Lastly, flat networks have weak security.

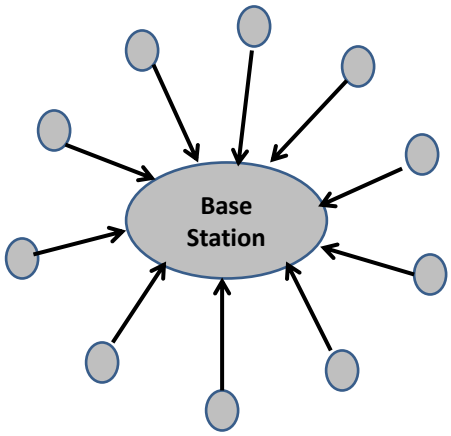

Fig. 2.1.: Flat Network

2.1.2 Cluster-Base Network

If we partition the network into regions or groups of nodes, then we call this a cluster-base network. A cluster-base network is more energy efficient than a flat network. In the cluster-base network there is no flooding through the base station. Cluster-base networks have many advantages [4]:

1. Data will not be flooded through the whole network.

2. The backbone of the network will not be complicated. It will be based on the number of clusters in the network. The network would be more manageable from a security point of view.

3. A change in a node will affect that cluster and not the entire network. For example, if a node becomes faulty or malicious it will not cause significant damages on the network. If a node moves it might move inside the cluster or it will become another cluster's member.

In most of the cluster-base networks there are three types of nodes [4]: *i*) *Cluster Heads (CH)*, *ii*) *Normal Nodes*, and *iii*) *Gateway Nodes*. The cluster head has an important role in a cluster, it will gather the data from the normal nodes in the cluster (aggregates it). Also she generates data the same as normal nodes. Normal nodes only generate data and gateway nodes are nodes that belong to more than one cluster and the existence of them are not mandatory for a cluster. In Fig. 2.2, we illustrate a cluster-base network.

The size of a cluster is an important parameter. If the size of a cluster is too large the communication between nodes would be complicated and uses too much energy (if the distance between nodes are large). Furthermore, if the size of the cluster is too small it makes the infrastructure of the network complicated (there would be a lot of clusters in the network). So there is a trade-off between the size of the clusters and the number of clusters.

If e is the power to transmit a message to another sensor with the distance of d, then the following models the power consumption e for sending the message [4, 5]:

$$e = kd^c. \tag{2.1}$$

In above equation k and c are constants based on the wireless system, usually $2 < c < 4$. Since the transmission power is at least the square of the distance between two nodes, we can save more energy if the nodes could send the data in a hierarchical fashion, which means using cluster-base network.

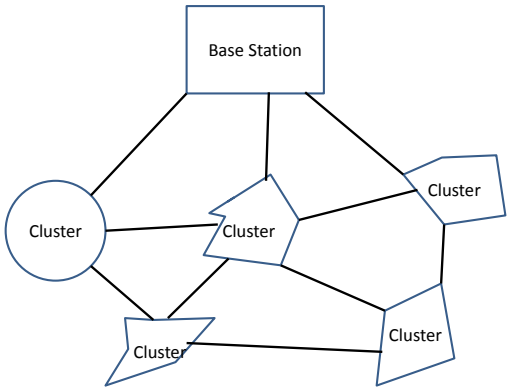

Fig. 2.2.: Cluster-Base Network

2.1.3 Sensor Network

In a network the sensor's life cycle would mainly consists of three phases: (i) the joining phase (a node would join a cluster), (ii) the stable phase (which means a node belongs to a cluster and has not changed her location yet), and (iii) the leaving phase (a node abandons her cluster and joins another cluster). Also, each node has a particular *authentication mode*: (i) a node could be malicious.(ii) a node has a lack of battery and becomes faulty, (ii) a node could be in a hibernate mode which means she cannot detect the malicious behavior if there is any, and (iv) an Honest node. We

consider hibernate nodes as honest nodes because if even a sensor is malicious when it is in hibernate mode it cannot do that much damage to the network.

There are several factors which impact the performance of a network and we need to consider them:

1. **Suitable density:** If a sensor field does not have suitable density, then there would be several clusters with few members. Nodes need to extend their transmission range which is expensive in the case of energy usage.

2. **Mobile/immobile:** If nodes in the network network are mobile, then cluster membership is dynamic. Sometimes they need to re-cluster.

3. **Battery power limited lifetime:** A node's energy is not unlimited and they are not connected to the renewable source of energy, therefore any node in a network could be faulty at anytime.

2.2 Network Security

There are two types of attacks that might cause damages in a network: **active** and **passive** attacks. **Active** attacks are the one that the adversary actually interferes with the network communication. For example, the adversary modifies data or causes disconnections in a route. **Passive** attacks are attacks where the adversary is not actively trying to interfere. For example, the adversary would listen to the network communication (eavesdropping) [2,6]. An ad-hoc network is always prone to a variety of attacks such as:

1. **Sybil Attack:** A node in the network can make different identities. Therefore that node can steal an honest node's identity (impersonate) to access the data [6]. Without Logically centralized authority, Sybil attacks are always possible.

2. **Denial of Service Attacks (DOS):** An attempt to disconnect the users from the network.

3. **Data Aggregation Attack:** Data in the wireless network can be **altered** [2,6]. Therefore the decision could be made based on the faulty or modified data that would be questionable.

2.3 Cryptography

The three aspects of security are commonly partitioned into three areas: Availability, Confidentiality and Integrity. There are numerous possible security services: such as privacy, authentication, etc. A security mechanism is a tool/technique that is used to provide a security service.

Cryptographic techniques are security mechanisms. There are numerous cryptographic techniques and they have been developed to provide some security service. For example, encryption is a security mechanism that is used to provide privacy.

A cryptosystem $(E, D, \mathcal{M}, \mathcal{C}, \mathcal{K}_E, \mathcal{K}_D)$ is a tuple which consists of an encryption function E, a decryption function D, a message space $\mathcal{M}$, an encryption keyspace $\mathcal{K}_E$ and an decryption keyspace $\mathcal{K}_D$ such that:

$$E : \mathcal{M} \times \mathcal{K}_E \to \mathcal{C},$$

$$D : \mathcal{C} \times \mathcal{K}_D \to \mathcal{M},$$

where $D_{k'}(E_k(M)) = M$.

Symmetric key cryptography, is a class of cryptosystems for which it is easy to compute the decryption key given the encryption key. Most symmetric key cryptosystems are such that the decryption key is the same as the encryption key. Suppose (E, D) represents a cryptosystem, where E is the encryption transformation and D is the decryption transformation then (E, D) is called a symmetric key cryptosystem if given the encryption key k_e it is computationally easy to compute the corresponding decryption key k_d.

There are two types of symmetric key algorithms [7],

1. **Stream Cipher:** successive plaintext elements are encrypted using the same key,
$$y = y_1 y_2 \cdots y_L = e_k(x_1)e_k(x_2)\cdots e_k(y_L). \tag{2.2}$$
2. **Block Cipher:** operates on fix length groups of bits called blocks.

Common block cipher algorithms are: AES, DES, and Triple DES. A common stream cipher is RC4 [8].

Some advantages of symmetric key cryptosystems: they tend to be fast, simple and uses less computational resources in comparison to public key cryptosystems. Symmetric key encryption requires a high level of trust and it is the major draw back of this cryptosystem. It requires a secure channel which parties can trust to exchange the symmetric key. Sharing a symmetric key initially is a problem, it has to be shared in a way that makes sure it remains secret during the exchange.

Asymmetric cryptography is also known as *public key cryptography*. This cryptosystem uses two keys: a public key and a private key. A public key can be made publicly available and is used to encrypt messages by anyone who wants to send a message to that person whom the key belongs to. Private keys need to be kept secret, and they will be used to decrypt messages.

Theorem 2.3.1 *Suppose* (E, D) *represents a cryptosystem. We call it an asymmetric key cryptosystem if it is not a symmetric key cryptosystem. Thus it must be computationally hard to find* d_k *given* e_k *decryption key given the encryption key [7].*

Some cryptosystems that are public key schemes includes: RSA, ElGamal (signature scheme).

The advantages for using a public key encryption are:

1. No need to have a key distribution center.

2. No need to have a secure channel to share a symmetric key. Each member in network will publish her public key and will keep her private key.

The disadvantages for a public key are:

1. The public key should be authenticated. Without authentication, the sender would not know the public key actually belongs to that specific sensor or not.

2. Speed. Public key encryption is significantly slower than symmetric key encryption.

3. Utilizes more computer resources. Public key requires more computational resources, such as memory and CPU time, in comparison to symmetric key encryption.

4. Loss of private key can cause severe security problems. An important problem is how to handle key update. If the private key is lost by a sensor, then all the messages that have been sent to that sensor can be read by an adversary.

2.3.1 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) has been applied for some time in wireless networks. The advantages of using elliptic curve cryptography are: (*i*) speed of elliptic curve decryption (and elliptic curve digital signatures) (*ii*) bandwidth, and (*iii*) less power and memory will be needed to do these computations (which is important in a wireless setting where the computing power, memory and battery life of nodes are limited). This allows a great saving in hardware implementation [9].

2.4 Cryptographic Hash Function

Cryptography hash functions play a fundamental role in modern cryptography [8]. Furthermore, hash functions also have been used in non cryptographic computer applications. The focus of this topic concerns message authentication.

Definition: A hash function h is a function that satisfies the following properties [8]:

1. **Compression:** h maps an input x for an arbitrary bit-length to an output $h(x)$ fixed bit-length.

2. **Ease of computation:** Given h and an input x it is computationally easy to calculate $h(x)$.

3. **Preimage resistance:** It is computationally hard to find any input which hashes to that output. If $h(x') = y$, given y the corresponding input is not known.

4. **2nd-preimage resistance:** It is computationally infeasible to find any second input that has the same output as any specified input. For example find the 2nd-preimage $x \neq x'$ such that $h(x) = h(x')$.

5. **Collision resistance:** it is computationally hard to find any two distinct inputs x, x' which hash to the same output such that $h(x) = h(x')$.

The typical usage of hash function in cryptography is as an integrity check. In this usage, the problem of determining the integrity of a large message is reduced to fixed-sized small hash values, i.e. a user can maintain a verification of an original message by computing a hashed value of the original and then anytime it reloads the data it can verify its integrity by comparing this to the hash of the original data. If the corresponding hash value from the current data was not the same as the original hash value then the data has been altered. MD5, SHA1 and SHA2 are commonly used cryptographic hash functions.

2.5 Digital Signature System

A digital signature scheme can be created by utilizing public key cryptography in a way that anyone in the network can verify the signature and only the holder of the private key can generate the signature. To design a signature scheme, it is necessary to make two algorithms [10] one for signing and the other one for verifying the signature. The verifying algorithm is assumed to be accessible to all potential receivers.

A digital signature would have the following features [8, 10]:

1. A plaintext message space.
2. A signature space (all possible signatures).
3. A key $K_1 \times K_2 \to K$.
4. A set of key generation algorithm that generates $(k_1, k_2) \in K_1 \times K_2$ where k_1 is the secret key and k_2 denotes the corresponding public key.
5. An efficient signing algorithm, $Sig : K_1 \times M \to S$ assigns a signature s to a pair: the secret key $d \in K_1$ of the signer and the message $m \in M$, i.e., $s = Sig(d, m) = Sig_d(m)$.
6. An efficient verification algorithm $Verify : S \times M \times K_2 \to \{true, false\}$.

For any secret $d \in K$ and any $m \in M$:

$$s = Sig_d(m), \tag{2.3}$$

The signature of m is denoted by s.

$$Verify_{p_k}(m, s) = \begin{cases} True & \text{if } s = Sig_d(m), \\ False & \text{if } s \neq Sig_d(m). \end{cases} \tag{2.4}$$

Here p_k denotes the corresponding public key. A digital signature system is sometimes constructed by using a public key encryption, like an RSA scheme or ElGamal scheme.

ElGamal Signature

In the ElGamal signature scheme, one works in a finite field Z_p^* (also known as $GF(p)^*$) where p is a suitable large prime. The secret key k belongs to Z_{p-1} and the corresponding public key is $g^k \bmod p$.
The user will publish g, p and g^k and will keep k secret.
Signing: To sign message $m \in GF(p)^*$ the user selects a random integer $r \in Z_p^*$ [10] where $gcd(r, p-1) = 1$ then she calculates:

$$x \equiv g^r \bmod p. \tag{2.5}$$

She will calculate:

$$m \equiv k \cdot x + r \cdot y \bmod p - 1. \tag{2.6}$$

The signature is:

$$s = Sig_k(m) = (x, y). \tag{2.7}$$

Verification: Given m and $\tilde{Sig} = (\tilde{x}, \tilde{y})$. The goal is to check if $\tilde{Sig}$ is a valid ElGamal signature of m:

$$VER(m, \tilde{Sig}) = \left(g^{\tilde{m}} \stackrel{?}{\equiv} (g^k)^{\tilde{x}} \cdot \tilde{x}^{\tilde{y}} \bmod p \right). \tag{2.8}$$

2.5.1 Threshold Signature

If a group consists of n users and they wish to have the ability to create a signature, as long as t many users agree to cooperate, and cannot generate a signature with less than t users cooperation, it is called a t out of n threshold signature scheme, (t, n). Here t is the threshold, the users are called shareholders. The data the shareholders use to construct a signature are called a share of the key. The entity that construct the shares is the dealer and the phase they use to construct the signature is the reconstruction phase [11].

Each participant P_i, $(i = 1, \cdots, n)$ in the group can construct a partial signature s_i for message m. They can partially sign a message:

$$Sig_{s_i}(m) = \textit{ partial signature.} \tag{2.9}$$

If threshold number of nodes in a cluster partially sign a message they can put the cluster signature on the message:

$$Sig_{sk}(m) = \sum_{P_i \in \mathcal{B}} a_i Sig_{s_i}(m). \tag{2.10}$$

where a_i is some publicly known constants dependent on $\mathcal{B}$.
In our work, the partial signature is important especially for signing messages that will be transmitted outside the cluster or if nodes want to exclude a node from the cluster. In Chapter 6, we will discuss the usage of this signature.

2.5.2 ElGamal Threshold Signature

The goal is to construct a t out of n ElGamal threshold signature scheme. Here any t participants will be able to partially sign a message, called a partial signature. Once we have t partial signatures we will be able to construct an ElGamal signature. Initialization [10]:

1. The dealer selects a collision resistance hash algorithm H, a prime modulus p, a prime q that divides $p-1$, and a generator $g \in GF(q)$. Also the dealer needs to choose a polynomial f of degree at most $(t-1)$. In addition, the dealer chooses a public element x_i for each participant $P_i \in \mathcal{P}$.

2. The dealer sends privately to each participant P_i a share $s_i = u_i + f(x_i)$ where $u_i \in_R GF(q) \backslash 0$, the key is $k = f(0)$ therefore the public key of the group is $y \equiv g^k$. Each participants has its own public key $y_i \equiv g^{s_i} \bmod p$ and $z_i \equiv g^{u_i} \bmod p$.

3. The dealer publishes (H, p, q, y) together with $\{(y_i, z_i) | P_i \in \mathcal{P}\}$.

Signing: Given message m and $\mathcal{B}$ a set of t participants:

1. Each participant $P_i \in \mathcal{B}$ chooses a secret key as $k_i \in Z_q$ and computes $r_i \equiv g^{k_i} \bmod p$. The value r_i will be broadcasted.

2. Each participant $P_i \in \mathcal{B}$ computes, for $\mathcal{B} \subset \mathcal{P}$, with $|\mathcal{B}| = t$:

$$\begin{aligned} R &= \prod_{P_i \in \mathcal{B}} r_i \equiv g^{\sum_{P_i \in \mathcal{B}} k_i} \bmod p, \\ E &\equiv H(m, R) \bmod q. \end{aligned} \tag{2.11}$$

3. Each participant P_i calculates their partial signature as:

$$c_i \equiv s_i \prod_{P_j \in B; j \neq i} \frac{-x_j}{x_i - x_j} + k_i E \bmod q. \tag{2.12}$$

   Then the participant will send (m, c_i) to the combiner.

4. Combiner verifies all the signature by using the following equation:

$$g^{c_i} \stackrel{?}{\equiv} y_i^{\prod_{P_j \in B; j \neq i} \frac{-x_j}{x_i - x_j}} . r_i^E \bmod p. \tag{2.13}$$

   then the combiner will compute:

$$\sigma = \sum_{P_i \in P} c_i \bmod p.$$

   The triple $(\mathcal{B}, R, \sigma)$ is the signature of m.

Verification: A verifier, of signature $(\mathcal{B}, \tilde{R}, \tilde{\sigma})$ for message m checks if:

$$g^{\tilde{\sigma}} \stackrel{?}{\equiv} \tilde{y} T \tilde{R}^{\tilde{E}}, \tag{2.14}$$

where T and E are:

$$T \equiv \prod_{P_i \in B} \tilde{z}_i^{\prod_{P_j \in \mathcal{B}; j \neq i} \frac{-x_j}{x_i - x_j} \cdot r_i^E} \bmod p, \tag{2.15}$$

$$\tilde{E} \equiv H(\tilde{m}, \tilde{R}) \bmod q. \tag{2.16}$$

If the check in Equation (2.14) is true then the signature will be accepted.

2.6 Group Conference Key

In [12] Burmester and Desmedt described several secure conference key distributions for a variety of network topologies. In our work, nodes are broadcasting their data inside the cluster, so we describe their broadcast scheme here. Let $P_1, \cdots, P_n$ be a set of users in a cluster, the users in a cluster will generate the cluster key based on following, here p and q are two large prime numbers where $q|p-1$:

1. Each P_i, $i = 1, \cdots, n$, selects $r_i \in_R Z_q$ and then computes and broadcasts $z_i = g^{r_i} \bmod p$.

2. Each user checks the order satisfies $ord(g) = q$. Then she computes and broadcasts:
$$X_i \equiv (z_{i+1}/z_{i-1})^{r_i} \bmod p. \tag{2.17}$$

3. Each user computes the conference key as follows:
$$K_i \equiv (z_{i-1})^{nr_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i-2} \bmod p. \tag{2.18}$$

Honest users compute the same key:

$$K \equiv g^{r_1 r_2 + r_2 r_3 + \cdots + r_n r_1} \bmod p. \tag{2.19}$$

In the key generation schemes that will be used, all the nodes need to be authenticated sequentially, if the node i could not pass the authentication process node $i+1$ will be halted.

3. RELATED WORKS

The purpose of our research is to secure the network in an energy efficient manner. To achieve this we need to have proper topology, as well as the proper choices of security and key sharing schemes. Several researchers have proposed different topologies and key sharing schemes, we introduce some of the key properties in this chapter.

As mentioned in Chapter 2, several different network topologies have been proposed. Because the flat network is not a proper architecture for mobile ad-hoc networks, researchers have mainly focused on the cluster-base network [4]. Some of the work in cluster-base networks are energy efficient schemes [13, 14], some are K-tree schemes [15–17]and some are management schemes [18–20].

Regarding the security of the network, some of the researchers have noted that nodes need to monitor each other in a network, which we call the "neighborhood watch" [21,22]. Marti, Giuli, Lai and Baker [23] call the nodes monitoring the "watchdog". In their research [23], if nodes do not hear forwarding packets from other nodes, they regard it as a malicious behavior. Nodes send their packet based on the path rating. A disadvantage of their work is that they do not punish/disconnect malicious nodes from network.

Buchegger and Le Boudee [24] proposed the CONFIDANT (Cooperation Of Nodes) scheme, which is similar to a node monitoring scheme in the network. In their approach, a node understands if the destination node received the message by receiving acknowledgment from the destination node. In their work, they isolate the malicious node from the network in a way that each node chooses a route that does not contain the malicious node.

In Rabinovich and Simon's [21] work, the network has been divided into regions and each region has its own server. The server decides which sensor is trustworthy or malicious. Reputation analysis is one of the responsibilities for the server. Furthermore, in their work, servers have vital role in the network, if server is not connected to a source of energy, her battery will drain quickly.

In all of the cluster-base networks, the cluster head consumes more power than the other nodes (due to the amount of communication). There are many different approaches to save energy for a cluster-base network for a node which is the cluster head. In the Lin and Liu [25] scheme, they rotate cluster head roles to other nodes. In [25], the authors proposed a way that each node communicates the amount of energy it has, then in the cluster they re-elect the cluster head based on the amount of energy. If a node has more energy than other nodes in a cluster she will be a cluster head. A problem with this approach is that if a node is malicious and wants to be a cluster head, she may lie about the amount of her energy to become a cluster head. Moreover, re-clustering may be unnecessary, which could be expensive in case of energy usage. There is another approach called, TDMA (Time Domain Multiple Access) [26] which can be used to reduce the amount of energy usage in the whole network. In this approach the node will go to sleep mode if that node is inactive (duty cycle).

Zhu, Setia and Jajodia [22], proposed Secure Deep Throat (SDT) protocol. In their proposal, nodes can use the concept of "witness anonymity" for peer-to-peer systems. The concept of "witness anonymity" has some disadvantages. First for honest nodes, second for malicious nodes who wants to use the anonymity system. With SDT, nodes can make claims against each other without the fear of retaliation by keeping the identity of a node who claimed against another node. Also it will determine a malicious node's identity if that node tries to misuse the anonymity (send multiple claims against an honest node). If all the adversaries collude together to find out the source of anonymous claim they would not find that out as long as the node would not send multiple claims against a node. The anonymity of a witness

could be maintained even if other members in the network are compromised at a later time. If a node sends multiple claims against another node, her identity would be revealed. In their work, they used a Mixnet-based [27] anonymous communication system so that if a node sends a claim no one can find out her identity. To send claims, each node maintains two claim databases, one is a private claim and the other is a common claim. If a node sees a malicious behavior she will store it in her private claim database, otherwise if the claim comes from another node she will send it to her common claim database. Each complaint consist of two IDs [21] : the suspect sensor's ID and the reporting sensor's ID. We used their proposal as a model for our claim broadcasting protocol with the difference of there is no anonymity in our network, also any anonymous claim would be considered a malicious behavior.

In the Bandyopadhyay and Coyle [28] proposal concerning "Hierarchical Clustering Algorithm", each sensor could be a cluster head with probability p. If the node is a cluster head, she will send advertisements to neighboring nodes for at most k hops away. The motivation of their proposal is to make the cluster based network more energy efficient. Moreover, in their proposal they assumed the base station was centrally located which is an ideal location for a network if nodes need a trusted third party but in practice a centrally located base station might not be possible. We adopt our network organization from their work with several modifications, which we discuss later in Section 6.3.1.

Because the number of nodes in a network is small and nodes do not have any renewable resources, they need to be organized in an energy efficient manner. According to Bandyopadhyay et.al. if the sensors distributed according a homogeneous spatial Poisson process, in a field with the side of $2a$ and the number of nodes in the area is random variable N with mean of λ. Then the field area is $A = 4a^2$. If we assume that the base station is in the middle of a square area with the amount of n nodes in the area. The probability of a node becoming a cluster head is p, therefore we have on average np cluster heads in the network. Let D_i be a random variable which denotes

the length of the segment from a sensor located at (x_i, y_i), where $i = 1, 2, \cdots, n$ to the processing center. Therefore, we have following as result:

$$E[D_i|N = n] = \int_A \frac{1}{4a^2}\sqrt{x_i^2 + y_i^2}dA = 0.765a. \tag{3.1}$$

Since the probability of a sensor becoming a cluster head is p, cluster heads and non-cluster heads are distributed based on the non independent spatial Poisson process with intensity of $\lambda_1 = p\lambda$ and $\lambda_0 = (1-p)\lambda$. As we will describe later, in the ideal setting the nodes in the network form a Voronoi Tessellation, and the zones in the plane called Voronoi cells. If N_v is the random variable denoting the amount of nodes in the Voronoi cells corresponding to the nucleus (cluster head) and L_v is the total length of the nodes in Voronoi cells connecting to the nucleus, then according to the [29] we have:

$$E[N_v|N = n] \approx E[N_v] = \frac{\lambda_0}{\lambda_1}, \tag{3.2}$$

$$E[L_v|N = n] \approx E[L_v] = \frac{\lambda_0}{2\lambda_1^{\frac{3}{2}}}. \tag{3.3}$$

As Bandyopadhyay,et.al [28] noted that the total energy that has been used in a Voronoi cell would be (C_1 is total energy):

$$E[C_1|N = n] = \frac{E[l_v|N = n]}{r}. \tag{3.4}$$

They also calculated the probability of being a cluster head in the network with an optimal usage of energy for invitation and based on that the optimal amount of hop number in a cluster. For sending the data to the base station also they used levels for cluster heads (level 1, level 2,$\cdots$), and for sending the data to cluster head level 1 sends it to level 2 and so forth to the base station. For a cluster head to be in a certain level she needs to flip a coin with some probability.

4. DATA AGGREGATION

Data aggregation means finalizing information based on other sensors' data. If a sensor or couple of sensors in a network are faulty then the decision made, based on that faulty data could be questionable. Simple statistical functions can be used for misleading data, for instance: approximate maximum, approximate minimum, approximate average, and approximate median [30]. In this chapter these statistical functions and their usage have been discussed.

4.1 Background About Statistical Functions

Consider in a sensor network n sensors collect data, data will be propagated to the base station. At base station data will be gathered (aggregated) and processed into information. This can be represented in some computation $f(x_1, x_2, \ldots, x_n)$, where the $x_1, x_2, .., x_n$ represent the sensor readings. Here f is some mapping $f : D_1 \times D_2 \times \cdots \times D_n \longrightarrow \Gamma$ which each D_i represents domain and also Γ represents all possible type of information.

$$y = f(x_1, x_2, \ldots, x_n). \tag{4.1}$$

By looking at the above equation, we observe that the sensor's reading will be turned into information and if one or some of the readings are faulty then the information resulting of these readings would be faulty. If, for example, the true reading of sensor 3 is x_3 and the false reading of sensor 3 is $\acute{x_3}$, then the information is $\acute{y} = f(x_1, x_2, \acute{x_3}, ..., x_n)$ so that $y \neq \acute{y}$ where y is the "true information", which is the information that would be the result of sending x_3 instead of $\acute{x_3}$.

We need to have some kind of metric to calculate the statistics.A necessary tool that is used to measure the quality of an approximation is some type of metric.

Formally a metric is some real-valued function ρ defined on some set $\mathcal{D} \times \mathcal{D}$ satisfying the following: (i) $\rho(x, y) \geq 0$; (ii) $\rho(x, y) = 0$ if and only if $x = y$; (iii) $\rho(x, y) = \rho(y, x)$; and (iv) $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$ [31] . Common choices of ρ will be the Euclidean distance $\rho(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$ and $\rho(x, y) = \max_i |x_i - y_i|$.

We will denote the metric/measurement between x and y as $|x - y| = \rho(x, y)$. In theory, one would want to use a metric to measure the quality of the approximation, but in practice it may not be a useful measure. Consider the following example [30].

Example 1 *Suppose n seismic sensors are collecting seismic readings. The sensor data is then processed to compute $y = f(x_1, \ldots, x_n)$ where $y = (y_1, y_2)$, here y_1 represents the time prediction of an earthquake and y_2 represents the duration of the earthquake. Suppose $y^* = (y_1^*, y_2^*)$, then to measure the quality we need to compute $|y - y^*|$, but observe that many of the common metrics fail to capture the essence of the prediction (it is more important to determine date than duration) [30].*

Therefore the quality of approximation should make sense. Also it is possible that the best measure does not posses all the properties of a metric.

4.2 Statistical Functions

Let's assume that we have n sensors, $S = \{s_1, s_2, \cdots, s_n\}$, if the expected number of faulty sensors are less than k. We characterize this as $(n, k) - property$, in a way that the true reading would not be distinguishable from the faulty reading. But if the readings are outside some possible bounds those readings would be removed from consideration. If, for example, one of the readings is out of bound we have the $(n - 1, k - 1) - Property$. Giving some sensor's reading it is *computationally* hard to decide if x is a "true reading" or not [32].

Faulty reading questions the security of the statistics which were introduced by Wagner [33]. The base station can compute some type of simple statistical functions to understand if the sensor's reading is out of bounds or not, these statistical functions include: maximum, minimum, mean, median and mode. A function $y = f(\cdot)$ is

insecure if $|y - y^*|$ is not suitably small, where y is true reading and y^* is faulty reading. The definition of "suitably small" will vary, dependent on the context of the application for which it is used.

The statistic *sum* is calculated as $sum = x_1 + x_2 + \cdots + x_n$ and in the presence of a faulty sensor it would be $sum^* = sum + (x_3^* - x_3)$. As you can see, if one reading has a huge error it can affect the sum completely.

Another statistical function is called *count* which has been introduced by Wagner [33]. In Wagner's work each sensor sends 1 or 0, so in the case of faulty/malicious readings, the *count* will not change significantly because if there is k faulty sensors, k is limited and *count* will change at most k. Therefore, we can consider that *count* is a secure reading.

The *average* is calculated as $avg = (x_1 + x_2 + \cdots + x_n)/n$. *Average* is insecure and this follows from the fact that *sum* is insecure, $avg = sum/n$. In the presence of a faulty sensor, we have $avg^* = (x_1 + x_2 + x_3^* \cdots + x_n)/n$ and $avg^* = avg + (x_3^* - x_3)/n$. Therefore, if just one of the sensors goes faulty the result could be significantly different from the true result.

If the base station calculates the minimum of a sensor's reading as $min\{x_1, x_2, \cdots, x_n\}$, then the attacker can completely control the result by significantly reducing one of the sensor's reading, so *min* is also insecure.

If the base station computes the maximum of sensor's reading as $max\{x_1, x_2, \cdots, x_n\}$, the attacher can control the result by increasing one of the sensor's reading significantly, also *max* is insecure.

Suppose the base station computes the *mode*, which is calculating the most frequent data. If there is k faulty sensors, they could impact the result. Therefore, *mode* is insecure.

4.3 Approximating Statistical Computations

In this section approximate calculation have been introduced to remove the reading which are out of bounds.

4.3.1 Approximating Minimum

How could the quality approximation be understood for the minimum? Why would anyone need to calculate the minimum? Is it because of minimum can affect the performance of a device? Or some action needs to be done? They will vary based on different situations, but probably the action would be based on the minimum in comparison to some bound, if the minimum value is below the bound an action will take place. If the sensors are faulty they could affect the minimum reading significantly and some action should not be invoked based on the faulty reading.

We would characterize the *best metric* for comparing some min^* (the approximation to minimum) to min (the true min) to satisfy:

$$min \leq min^*. \tag{4.2}$$

Given $S = \{s_1, s_2, \cdots, s_n\}$ the base station can sort the data, outputting it as:

$$x_1 \leq x_2 \leq \cdots \leq x_n. \tag{4.3}$$

Our definition of min^* satisfies:

$$min^* = (k+1)^{st} smallest = x_{k+1} \tag{4.4}$$

The property of min^* satisfies:

Theorem 4.3.1 *The approximation min^* satisfies $min^* \geq min$ (here min is the minimum of the "true readings").*

Proof. Let S represent n sensor reading satisfying the $(n,k) - property$. Let $x_1, x_2, \ldots, x_n$ denote the sorted data S. Now $min^* = x_{k+1}$. Consider the set

$x_1, x_2, \ldots, x_{k+1}$ these are k readings (in increasing order), so at least one of them is a true reading. Then $min^* \geq$ " true reading" $\geq min$.

We can generalize the question "calculate the *min*", to questions like calculate the $2^{nd}min$, calculate the $3^{rd}min$, ... In general the $i^{th}min$, is the i^{th} smallest "true data value", in the presence of at most k faulty sensors readings. We suggest an approximation for $i^{th}min$ as:

$$i^{th}min^* = (k+i)^{st} \text{ smallest } = x_{k+i}. \tag{4.5}$$

Then, $i^{th}min \leq i^{th}min^*$, for $i = 1, 2, ..., n-k$.

4.3.2 Approximating Maximum

Why would someone calculate the maximum? Is it because of the performance of some device would be in danger? Or some actions need to be performed based on the maximum data? Probably the action would be based on some maximum bounds, for instance if the maximum would be above the bound, then an action needs to take place but the action should not be based on the faulty reading. The *best metric* we would characterize for some max^* (approximate max) would be less than or equal to the true maximum:

$$max^* \leq max.$$

We define max^* as:

$$max^* = (k+1)^{st} \text{ largest } = x_{n-k}. \tag{4.6}$$

Here the x_i represents the sorted data of all the sensors' reading (includes both true and false data). Then max^* satisfies $max^* \leq max$. Analogously, we can define the $i^{th}max$. We would approximate it as $i^{th}max^* = x_{n-k-(i-1)}$.

4.3.3 Approximating Average

The question is, what is the best approximation for an average in the presence of faulty data? Clearly the approximation should possess some attributes that an average satisfies. We first observe that:

$$\sum_{i=1}^{n-2k} i^{th}min \leq \sum_{i=1}^{n-2k} i^{th}min^* = \sum_{i=1}^{n-2k} x_{k+i}. \tag{4.7}$$

Next we observe that:

$$\sum_{i=1}^{n-2k} i^{th}max \geq \sum_{i=1}^{n-2k} i^{th}max^* = \sum_{i=1}^{n-2k} x_{n-k-(i-1)}. \tag{4.8}$$

Now observe that:

$$\sum_{i=1}^{n-2k} x_{k+i} = \sum_{i=1}^{n-2k} x_{n-k-(i-1)}. \tag{4.9}$$

Now $(\sum_{i=1}^{n-2k} x_{k+i})/(n-2k)$ is the average of the $n-2k$ smallest "true sensor readings". Further $(\sum_{i=1}^{n-2k} x_{n-k-(i-1)})/(n-2k)$ is the average of the $n-2k$ largest "true sensor readings". In order to approximate the avg, we define avg^* (our approximation to average) as:

$$avg^* = \frac{\sum_{i=1}^{n-2k} x_{k+i}}{n-2k}.$$

Then we have:

Theorem 4.3.2 *The approximation avg^* satisfies that it is greater than or equal to the average of $n-2k$ smallest "true sensor readings" and avg^* is less than or equal to the average of $n-2k$ largest "true sensor readings".*

The proof follows from applying Equations (4.7). (4.8), and (4.9).

4.3.4 Approximating Median

The property concerning the median: a median is a member for which roughly half of the members are smaller than or equal to and half of the members are greater than or equal to. In general, a set may have one or two medians (depending on odd

or even cardinality). In the presence of faults, it would be difficult to ascertain a correct median. If we have n sensors and k of them are faulty what we can ascertain is that $min \leq x_{k+1}$ and $x_{n-k} \leq max$. Thus we could measure a potential place for the true median by starting at these values and go right and left respectively by $n/2$ places (as illustrated in Fig. 4.1).

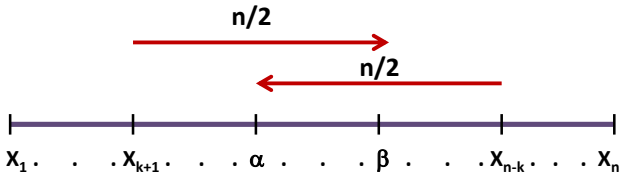

Fig. 4.1.: Approximate Median

We define $median^*$ as (α, β) where:

$\alpha = x_{n-k-\frac{n}{2}}$,

and

$\beta = x_{k+1+\frac{n}{2}}$.

That is, we define our approximation for median as an interval (α, β).

The $median^*$ satisfies the following:

Theorem 4.3.3

$$\alpha \leq median \leq \beta, \tag{4.10}$$

where $median^* = (\alpha,\ \beta)$ *and median is the "true median".*

Proof. Recall $min \leq x_{k+1}$, then $\beta = x_{k+1+\frac{n}{2}}$ is such that there are at least $\frac{n}{2}$ true data items less than or equal to β. The true median will have at least half of the true data items (which is bounded by $(n-k)/2$ and $n/2$). Thus the true median is less than or equal to β.

Similarly we can argue that the true median is greater than or equal to α.

5. SECURITY DESIGN

Several cluster base network architectures have been proposed [13–17, 25]. The majority of these cluster based network proposals have focused on energy efficiency. For example, by having the cluster head rotate or by having an efficient cluster size. However, in all of these proposed algorithms the cluster head plays an active role. Further, many of these algorithms require a significant amount of communication due to re-cluster. Note that due to re-clustering, nodes might drop packets and cause more communication problems. This problem occurs, because the cluster head plays an active role.

In our work, we reduce the amount of cluster related communication by limiting the cluster head to have more passive role. Thus, reducing the amount of re-clustering. We propose a scheme that distributes the work to all of the nodes in a cluster. Therefore the energy in the cluster head will not diminish as quickly as other proposed cluster base networks. Moreover, in a cluster base network with an active cluster-head if the cluster-head becomes faulty or malicious, the security and integrity of data in the network would be questionable. In our work, if a cluster head becomes faulty, other nodes in a cluster can detect and expel the node (cluster head) from the cluster. Therefore the cluster can continue its normal activity without having any problems caused by a faulty cluster head. In addition, in our work we do not use gateway nodes. Recall gateway nodes belong to multiple clusters, for integrity purposes we prefer to limit cluster accessibility for a given node.

5.1 Neighborhood Watch

As Balzanoand and Srivastava [34] noted, cryptography alone is not enough to secure a sensor network. A node can be compromised and cryptographic techniques

cannot distinguish the compromised sensor from an honest sensor [21]. We decided to make nodes monitor each other's activity, i.e. "police each other'', which we call "*neighborhood watch*". Rabinovish and Simon [21] also used the "*neighborhood watch*" for securing the sensor network. As observed in [30, 33], sensors can identify outliers, thus detecting false readings and/or faulty data aggregation.

The size of a cluster in a network is important. If it is too large, nodes cannot monitor each others behavior, at least not efficiently. If it is too small, malicious nodes can take over a cluster. We prefer to have clusters where nodes can police other. In the case of malicious behavior, other nodes can send a claim against the malicious node which we call "claim broadcasting".

As illustrated in Fig.5.1, nodes in a cluster monitor each other. Here "arrows" indicate that the node is within hearing range. For simplicity we did not show all the arrows.

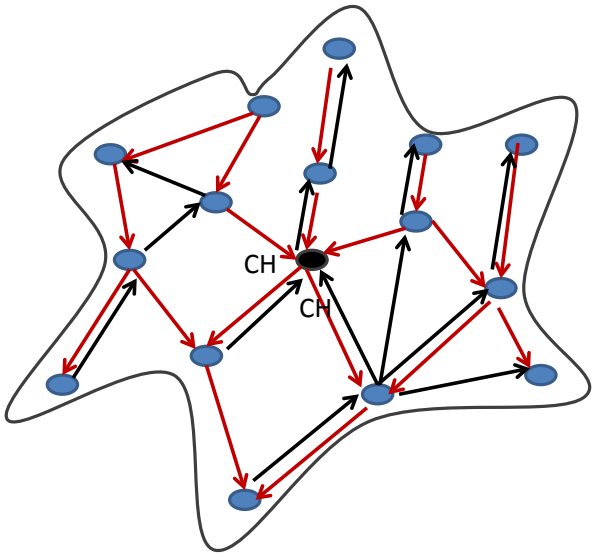

Fig. 5.1.: Neighborhood Watch

5.1.1 Claim Broadcasting

We modified the Zhu, Setia and Jajodia scheme [22] to construct our claim broadcasting scheme. Here, each sensor monitors the behavior of the other sensors in the cluster. To achieve this, each sensor has a claim database. If a node sees malicious behavior, she saves it in her claim database and broadcasts it to other sensors in the cluster. Once a majority of the sensors make a claim, they can punish the offending sensor.

In a sensor network, claims against a node could be troublesome as Zhu, Setia and Jajodia observed in their paper [22], if an honest sensor makes a claim against another node who is malicious, the malicious node may retaliate and make a claim against the honest node (a form of tit-for-tat behavior), for this reason they used witness anonymity scheme. In our network, we do not allow any anonymous claims and nodes would count the anonymous claims as malicious behavior.

5.2 Which Node is Malicious

There are two types of adversaries [22]: *selfish users* and *malicious users*. A selfish user may not participate/contribute to the cluster infrastructure and takes advantage of the cluster computing resources/network. For example, a selfish user might not forward a message have been sent to her. Malicious user is a user who will act mischievously for example they may want to modify the data or replay messages. Nodes may be malicious due to faulty behavior. For example, they may be faulty due to lack of energy sources such as diminished battery.

5.2.1 Detecting a Malicious Node

Not only is the detection of a malicious and/or selfish node in a network important, but discovery in a timely manner is vital. As [21] noted, "the more neighbors a compromised sensor has, the more complaints will be sent to the server and the

sooner it will decide to exclude that sensor". In our work, we do not have a trusted third party in a cluster, but we trust the majority of cluster members. Note, if a node is faulty they may unintentionally help malicious nodes. Observe, in our view of faulty node it is malicious.

5.3 Final Design

In summary the design should have these following qualities:

1. No network infrastructure.
2. In order to achieve greater integrity of data collection, we should be able to monitor the behavior of sensors in a neighborhood (cluster) at a low energy cost.
3. The network should be partitioned into small clusters.
4. There will be no active administrative head in each cluster.
5. We trust the majority.
6. There should be a mechanism to achieve a secure claim broadcasting.
7. The base station should be confident that the data they are using is accurate and trustworthy.

6. OUR APPROACH

In our work, nodes join a cluster, the nodes make keys autonomously, which means without any help from base station or a trusted central authority. Besides making a sensor network more organized, our network will be secured in an energy efficient manner. Some of the challenges include: How can we trust their identity? Are these sensors from the same network or not? We know that any sensor in a network can masquerade about her identity. If a node is malicious she can steal other honest nodes' identity and it makes them look malicious. One question is how in a network we can prevent the malicious or selfish behavior? If it happens how could the network detect that? In other words, how can we make a trustworthy network? In all wireless sensor networks we need to prolong the network's life time as much as possible with the suitable amount of security, therefore energy efficiency is another issue.

In this chapter, we provide the details about achieving the security design characterized as a "neighborhood watch" and we discuss different types of security schemes that we can use for our network to make the network more secure. By the end of this chapter, we will have answered all of above questions.

6.1 Assumption Concerning the Sensors that are Deployed

Trust is a significant barrier in our network, no nodes can trust each other when they first meet. In our work, we make the node's manufacturer as an off-line trusted party, much like a certificate authority [8]. The manufacturer can give nodes some information which can help the nodes trust each other. When a node is made by a manufacturer, the manufacturer will give the node an ID, a public key/private key pair and a manufactured date. All the information that the manufacturer provides for the node will be signed by manufacturer. This information, "ID Card", would be

comparable to a "Public Key Certificate". Whenever a node wants to communicate with another, she should transmit her "ID Card" to the other nodes (private key never leaves the node). Using signed IDs for all the nodes in the network would remove the problem of impersonation in the network. Fig. 6.1 illustrates the node's ID from manufacturer the dotted line shows the nodes private key. Note the private-key is not part of the ID. Rather it is stored in a secure location on the node.

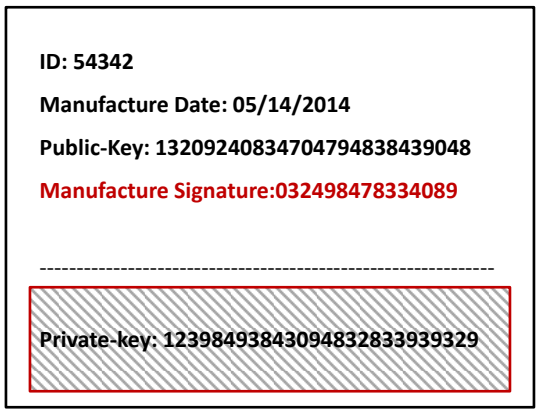

Fig. 6.1.: Node's ID from manufacturer

We assume all nodes are familiar with the manufacturer's public key and that nodes can distinguish the manufacturer's signature from other signatures. As a node ages, the trust of a manufacturer's signature will decrease. The longer a node is active the probability of being malicious becomes higher, especially since a node can become faulty due to diminished battery. Essentially, we have a situation of a certificate authority who is unable to revoke certificates. Thus the trust in the certificate will diminish over time.

In our model, we assume that the base station is located far from the sensor nodes and is not centrally located. The communications between nodes and the base station will be based on their job description, delivering the sensed data that is needed. Nodes do not contact the base station for superfluous reasons.

We achieve a cluster-base network that is more secure because of sensors visibility monitoring each other in a cluster. Each sensor knows their group members. Because of this, data is more trustworthy.

6.2 The Importance of the Manufacturer Information and Signature

Without having any central authority for which sensors can trust and rely on, some specific information cannot be trustworthy. Thus we need to have some kind of information which has been set by the manufacturer. Without the manufacturer signature, a node can impersonate and no nodes would ever know if the node she tries to communicate with is in her network or not. The list of information $\mathcal{I}_{P_j}$ that a node P_j needs from their manufacturer is:

1. Identification Number or ID.
2. Public Key $pk_{P_j,Manufacturer}$.
3. Manufactured Date.

Also all of these information have been signed by manufacturer, by:

$$\sigma_{P_j} = \text{signed ID} = Sig_{pk,Manufacturer}(\mathcal{I}_{P_j}). \tag{6.1}$$

6.3 Invitation Rules

In our work, nodes generate clusters autonomously. The cluster generation is executed in a greedy manner. A node accepts the the first cluster invitation that it receives and does not search for an invitation for which the cluster head is the closest. This is to ensure the communication links in the cluster does not become severed. In Chapter7, we discuss our simulation results of our approach and how closely it compares to the ideal cluster formation.

In our approach, each node has a probabilistic chance to be a cluster head. If they are a cluster head, they broadcast an invitation to other nodes to join their cluster.

Nodes that receive the invitation have the choice to accept or reject. It is possible that nodes may propagate the invitation to neighboring nodes depending on the preselected parameter *diameter* (hop size). These properties can be troublesome, you could possibly have neighbors for which one accepts and the other rejects causing a disconnection in the network. The following is the procedure for sending an invitation and making a cluster. We modified the Bandyopadhyay and Coyle [28] proposal:

1. Each node flips a coin with probability of p to determine if it should be a cluster head. This probability p is a preselected parameter. It is probably selected based on the density of the network and type of network security you want to achieve.

2. If a node is a cluster head, she will send an invitation to neighboring nodes with her "ID Card" from a manufacturer. In the invitation it will indicate how many hops it should be propagated, i.e. the desired diameter of the cluster, which we denoted by k hops.

3. Nodes listen and wait for the preselected period of time T.

4. If a node receives an invitation from another node(propagator), she will keep track of the sender(propagator) and the cluster head's information such as: the cluster head's ID, the number of hops, and her public key.

5. If a node, after expiration of time T, has not received any invitations, she will become her own cluster. That is, she is a "forced cluster head".

6. If a node gets an invitation, she will respond to the invitation immediately from a propagator.

   (a) The node sends to propagator, signing its acceptance and forwards its manufacturer certificate. This will be propagated all the way to the cluster head inviter.

7. After time T, a node who has decided to be in a cluster, will respond back to the propagator with her "ID Card" and a signed acceptance. This is continued to be passed back to the cluster head.

8. Once the cluster head gets a node's acceptance, she will verify all signatures and send the list of the cluster members their "ID Card" and then signs the list to all members. This list contains all the node IDs and public keys.

9. The cluster head sends a request to all the nodes in the cluster to make a group signature key and the cluster certificate.

Note, if a node accepts then the propagator accepts. This approach is a greedy approach "accept the first invitation".

All the nodes need to wait until time T to respond back to the cluster head but if all nodes respond back at the same time collisions may occur. This problem can be resolved by using different sized contention windows, this topic is out of the scope of this research.

6.3.1 Invitation Protocol

Recall $Sig_{sk,Manufacturer}$ denotes manufacturer's signature, CH denotes the cluster head and $\mathcal{N}$ represents the set of nodes in the network.

Algorithm 1 Invitation Protocol

1: **for all** Node $\in \mathcal{N}$ **do**
2: Node.CH=CoinFlip()
3: Run ClusterInvitation()
4: **for all** Node $\in \mathcal{N}$ **do**
5: **if** Node.$CH = Yes$ **then**
6: Send a list of all the nodes who accepts the invitation with her $Sig_{CH}(CH.ID\ ,\ CH.Pub_{Key})$
7: Send a request to make a group key and signature

Algorithm 2 CoinFlip()

1: $result \leftarrow$ Flip a coin //note this is a random value in the interval $(0, 1)$
2: **if** $result < p$ **then**
3: Node.CH=Yes
4: **else**
5: Node.CH=No
6: **return** Node.CH

Algorithm 3 ClusterInvitation()

1: **for all** Node $\in \mathcal{N}$ **do**
2: Node.$hop = 0$
3: Node.$inviter = 0$
4: **for all** Node $\in \mathcal{N}$ **do**
5: **if** Node.$CH = yes$ **then**
6: Broadcast invitation
7: **for all** nodes in the hearing range **do**
8: **if** Node hears Node$'$ AND Node.$inviter = 0$ **then**
9: Node.$inviter$ = Node$'$ information
10: **else**
11: **if** Node.$inviter \neq 0$ **AND** Node.$inviter.hop + 1 < K$ **then**
12: Node.hop = Node.$inviter.hop + 1$
13: Broadcast invitation with Node.$Sig_{sk,Manufacturer}$(Node.Pub_{Key} , Node.ID), nodes in the hearing $\rightarrow$ Node.$inviter = 1$
14: **for all** Node $\in N$ **do**
15: **if** Node.$inviter = 0$ **then**
16: Node.CH=Yes
17: **for all** Node $\in N$ **do**
18: **if** Node.$inviter = 0$ **then**
19: Node.CH=Yes
20: **else**
21: Send the acceptance to Node.$inviter$ with

Node.$Sig_{pk,Manufacturer}$(Node.Pub_{Key}, Node.ID)

The diameter of a cluster is dynamic due to the mobility of nodes. After some time, some nodes may leave the cluster. As illustrated in Fig. 6.2, a sensor receives and then propagates the invitation based on the current hop number and max-hop

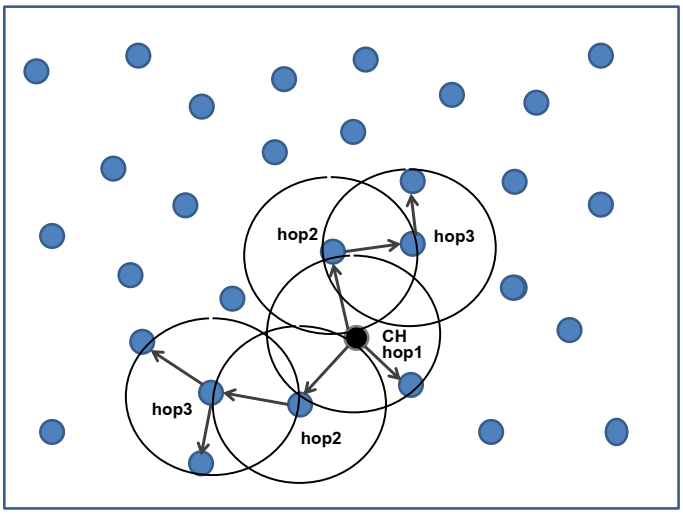

Fig. 6.2.: Invitation

parameter k. Here, circles around the nodes illustrate the broadcast transmission range and the arrow illustrates invitation propagation.

The concept of an ideal cluster formation is a Voronoi cell.

Definition 1 *[35] Let P be a set of points in $\mathbb{R}^2$ plane, a Voronoi diagram of $\mathcal{P}$, denoted by V_P, is the collection of Voronoi cells V_p. For each point $p \in \mathcal{P}$:*

$$V_p = \{x \in \mathbb{R}^2 | \|x - p\| \leq \|x - q\| \text{ for any } q \in \mathcal{P}\}$$

Fig. 6.3 [35] illustrates an ideal Voronoi cell. Here the dots represents the generator, in our case cluster heads. A node belongs to a cluster provided the distance between that node and its cluster head is smaller than the node to all other clusters.

The invitation algorithm has the same effect of Voronoi diagram, we simulated our invitation protocol using 50 nodes in an 8 by 8 region, each node has a range of 1.79, with the hop number of 2 and the probability of being a cluster head is 0.2, our results are illustrated in Fig. 6.4. CHs on the figure are cluster heads also nodes will wait till time T to get the best invitation with the least distance. Nodes will accept the best invitation in this simulation. In our test, nine clusters have been generated.

Our result is near to Voronoi Tessellation, but the result in this simulation is not realistic because for simulating the Fig. 6.4 we used C programming which is a serial program, because of that it is not a realistic result. To achieve a more realistic result, we simulated our algorithm in a threaded program. We will describe the simulation results in Chapter 7. In our work, the most important thing for making a cluster is visibility and number of nodes in a cluster, we are not worried about making a perfect Voronoi Tessellation, the reason we are using an algorithm to make a cluster near to a Voronoi cell is because the energy usage for nodes to communicate inside the cluster would decrease.

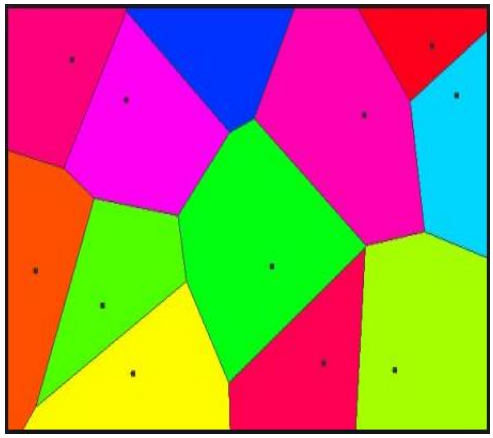

Fig. 6.3.: Voronoi Cells

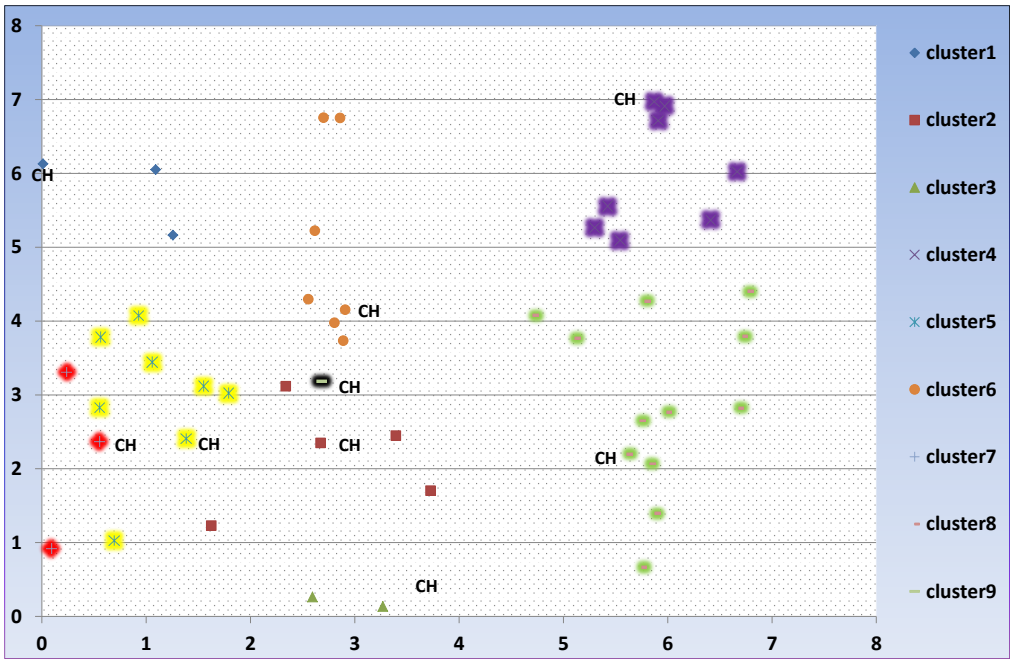

Fig. 6.4.: Tested Result

6.4 Cluster Head's Roles

In most of the cluster base networks, cluster heads play a vital, active role. As we have noted, in our work the role of a cluster head is limited. After a node decides to be a cluster head, she will send an invitation. Upon receipt of an acceptance, she will send the cluster list to all members. She then directs the cluster to generate a signature key. By constructing a cluster head with a limited role we are able to handle the case of a cluster head becoming faulty. Also the cluster head is the first node in a cluster who sets the symmetric key to nodes to communicate secretly inside cluster. We assume cluster heads are honest in the beginning of the formation of network.

6.5 Intra Cluster Communication

Within the cluster, private communication could be achieved by using public key cryptography, which is expensive. Also it could be based on a symmetric key

cryptography, which is easier and has less computation. A cluster can use a symmetric key for a period of time, after a while the cluster needs to change this symmetric key. The greatest draw back for using the symmetric key is it needs to be distributed over a secure channel which we can make the key passing secure by the following:

$$E_{P_{kP_i,manu}}\{Sym_{session}\}. \tag{6.2}$$

Initially the cluster head makes the symmetric key because she wants to communicate with every one secretly for sending the group information. Therefore, the cost for the cluster head to broadcast to all nodes k hop away would be at most kn. In the future, a node wants to communicate a symmetric key, the cost for her to send the symmetric key to all the nodes would be at most $2kn$.

6.6 Threshold Key Generation

The concept of group oriented cryptography has been introduced by Desmedt [36] in the way that when the key is controlled by a group (threshold of the group) rather than individuals. The following is Pedersen's scheme [37] on threshold sharing with verification, an improvement of Frankel and Desmedt threshold cryptosystem scheme [38]. Assume p, q and g are agreed in the network beforehand, such that p and q are two sufficiently large prime numbers which q divides $p-1$,G_q is a unique subgroup of Z_p^* of order q, and g is a generator of G_q.

Fix a cluster $\mathcal{C}$. If we have n sensors in cluster $\mathcal{C}$ and t is a threshold such that $1 \leq t \leq n$, then the key is shared to the n sensors so that t of nodes need to cooperate to generate the key. Also, we assume that $t \leq \frac{n+1}{2}$, in this assumption a majority of the sensors are honest sensors. The manufacturer has prescribed the p, q and g. The goal is for the cluster to generate a cluster public key $h \in G_q$, so that any t members can compute the secret key is x where $h = g^x \bmod p$.

We represent sensor i by P_i. We let $C(m,r)$ denote the commitment to $m \in \{0,1\}^*$, using the random stream of r.

The key generation is as follows:

1. P_i chooses a random number $x_i \in Z_q$ and computes $h_i = g^{x_i} \bmod p$. Then the sensor will select a random number r_i and broadcast the commitment $C_i = C(h_i, r_i)$ to all the members in the cluster.

2. When all n members of the cluster have broadcasted its commitment, then each node P_i opens the commitment $C_i = C(h_i, r_i)$.

3. Public key for the group is $h = \prod_{i=1}^{n} h_i = g^{\sum_{i=1}^{n} x_i}$, which is broadcasted to the cluster. The secret key is $\sum_{i=1}^{n} x_i$, notice no members know the key, but all know the public-key h.

   The following method shows that how x, which is the secret, is shared to the sensors. The degree of the polynomial is at most $t-1$.

   (a) Each sensor P_i chooses a random polynomial which has at most degree $t-1$ where $f_i(0) = x_i$ the polynomial would be:

   $$f_i(z) = f_{i_0} + f_{i_1} z + ... + f_{i_{t-1}} z^{t-1} \quad \text{where } f_{i_0} = x_i. \tag{6.3}$$

   (b) P_i computes $M_{ij} = g^{f_{ij}}$ which is the generator to the power of that sensors polynomial number. So $j = 0, ..., t-1$, Note $M_{i0} = g^{x_i}$ has already been transmitted.

   (c) When all sensors P_i in the cluster sent their t values, then P_i will send $s_{ij} = f_i(j)$ secretly and a signature on s_{ij} to P_j for $j = 1, ..., n$.

   (d) When all the sensors send those $t-1$ values, P_i sends $s_{ij} = f_i(j)$ secretly and a signature on s_{ij} to P_j for $j = 1, ..., n$.

   (e) P_i can verify the share it receives from P_j (which is s_{ji}) is consistent by the following equation:

   $$g^{s_{ji}} \stackrel{?}{=} \prod_{l=0}^{t-1} M_{jl}^{il}. \tag{6.4}$$

   If this fails P_i has discovered an error and terminates.

(f) P_i computes its share of the key by computing $s_i \sum_{j=1}^{n} s_{ij}$. Finally P_i signs h.

In the above scheme, if the amount of nodes in a cluster is n, in any broadcast a node's broadcast cost would be at most $2k$ because of the distance between two nodes would be at most $2k$. Because there are n nodes in a cluster, the cost of broadcast for all the nodes in a cluster would be at most $2kn^2$. In the above scheme for making a threshold signature in a cluster, there are three broadcasts, first for sending the commitment to all the nodes. Second, sending M_{ij}, And third sending the s_{ij} to all the nodes in a cluster. The cost of broadcast for M_{ij} would be at most tkn^2 and for sending s_{ij} is $2kn^2$. Therefore, the cost of making a threshold signature in a cluster would be at most $4kn^2 + tkn^2 = O(tn^2)$ and because number of nodes in a cluster are small the cost of broadcast is not significant. If we have a star base network, which means that all the nodes are one hop away from the cluster head, the cost will be calculated with $k = 1$.

6.6.1 Giving Access to a New Node in a Cluster

If a node joins a cluster then there should be some probation time for which activities can be monitored before the node is provided full cluster privileges, in particular shares of the signing key. The probation time will be determined based on if there exists a signed certificate membership from the previous cluster the node belonged to. If there does not exist a signed certificate of membership then the time will be longer in duration. The probation time can be kept by one of the members of the current cluster. The probation time will start once all members of the cluster have a copy of the node's public key and signed ID (by the manufacturer).

When the trust issue has been resolved group members will give the new group member permanent access, which means that the new node can participate in the group signature generation.

If the current cluster consists of members $\{P_1, \ldots, P_n\}$ and the new node is represented P_{n+1}. For $i = 1, \ldots, n$ let $f_i(z)$ denote the sharing polynomial (see Equation 6.3), the polynomial of degree $t-1$. Then cluster member P_i sends $M_{ij} = g^{f_{ij}}$ $(j = 0, \ldots, t-1)$ to P_{n+1}. In addition P_i sends $s_{i,n+1} = f_i(n+1)$ privately to P_{n+1}. Thus, P_{n+1} can verify the shares $s_{i,n+1}$ using $M_{i,j}$. Then P_{n+1} computes s_{n+1} by:

$$s_{n+1} = \sum_{j=1}^{n} s_{n+1,j}. \tag{6.5}$$

When node P_{n+1} wants to join a cluster, she first needs to broadcast her "Trust Certificate" from her previous cluster to the new cluster, the cost of this would be at most kn. After the probation time for the new node has expired, the new cluster generates for node P_{n+1} a share of cluster signature key using Equation (6.5). The cost for the broadcast needed in this procedure would be $2kn$. Therefore giving access to a new node P_{n+1} in a cluster would be $O(tn)$.

6.7 Sensor Leaving a Cluster

When a node leaves a cluster she should have her current cluster group members sign her trust certificate, if her trust rate is low she might not be able to immediately join another cluster.

How the cluster deals with a node leaving If a node leaves a cluster she will have her last share from the last cluster. After she joins another cluster she will eventually be provided a new share. However, if she increases her reception range she may be able to listen to the previous cluster's communication. There are several ways to handle this problem. First, because all the nodes have a claim database if a node leaves a group, then they can look at their own database and compute the trust rating of the node. If they see the node was trustworthy, then they may not reshare the key. Second, if we want to maintain high security, then immediately after a node leaves a cluster, the cluster will change the reshare the group key. Therefore other nodes which have left the group cannot hear any communication from their previous cluster. While more secure, this option consumes more energy. Between

these two options, there is a third alternative, a trade-off between the energy and security. Changing shares consumes energy and makes the network more busy and might cause some delays on data transaction, but if the shares do not change then there is a possibility that some node that is faulty or malicious can eavesdrop on communications. Definitely, if the threshold amount of sensors leave a cluster, the cluster needs to make new shares anyway due to lack of sensors to make group key. In practice, it is wiser to re-share whenever more than one-half the threshold has left.

6.7.1 Proactive Secret Sharing

In our work, we prefer to limit the amount of cluster key renewal, instead renewing the shares for the same key. If clusters change their key they need to renew their "cluster certificate" and this will require bandwidth. Herzberg et al. [39] proposed a *proactive* secret sharing scheme, which provides a way for the group key to stay the same but nodes in the network refresh (change) shares. The secret sharing scheme is based on Shamir's scheme [40], in their model there is a dealer who knows the secret key for the group, the dealer will choose a random polynomial of degree $t-1$ over Z_q subject to the condition $f(0) = x$ which x is the key for the group. Then the dealer calculates s_i for each participant and sends it to participants P_i. The dealer chooses a polynomial with degree $t-1$ over Z_q the same as following:

$$g^{s_i} = (g^{f_0})(g^{f_1})^i(g^{f_2})^{i^2} \cdots (g^{f_{t-1}})^{i^{t-1}} \mod p. \tag{6.6}$$

So if t sensors provide their shares they can compute the secret key x. Herzberg et al. assumed that there is a global clock that all the nodes can access and the shares would be changed after a period of time. They have t out of n threshold scheme, so that the adversary needs to at least compromise t nodes to access the key. If the shares do not change after some period of time, an adversary can get into system by using sufficiently many compromised nodes' shares. To defeat this, nodes in the cluster need to *renew* their shares after a period of time. Each sensor P_i will choose

a random polynomial δ_i of degree $t-1$ where $\delta_i(0) = 0$. So for each participant P_i we have:

$$f_{i,new}(z) = f_{i,prev}(z) + \delta_i(z) \bmod q. \tag{6.7}$$

Thus,

$$f_{new}(z) = \sum_{i=1}^{n} f_{i,new}(z) \bmod q. \tag{6.8}$$

Hence,

$$s_{i,new} = f_{new}(i) = \sum_{j=1}^{n} f_{j,new}(i) = \sum_{j=1}^{n}(f_{j,prev}(i) + \delta_j(i)) = s_{i,prev} + \sum_{j=1}^{n} \delta_j(i) \bmod q. \tag{6.9}$$

Because $\delta_i(0) = 0$, the secret x will stay the same and shares are going to be renewed. Let $a_1, a_2, \cdots, a_t$ be the interpolation coefficients, such that, based on Shamir's scheme $\sum_{i=1}^{t} a_i x_i$, nodes could get the secret key x from the following equation and the renewed share would not change the secret key x:

$$\begin{aligned} x &= \sum_{i=1}^{t} a_i s_{i(new)} && (6.10)\\ &= \sum_{i=1}^{t} a_i \left(s_i + \sum_{j=1}^{n} \delta_j(i) \right) && (6.11)\\ &= \sum_{i=1}^{t} a_i s_i + \sum_{j=1}^{n} \sum_{i=1}^{t} a_i \delta_j(i) && (6.12)\\ &= x + \sum_{j=1}^{n} \delta_j(0). && (6.13)\\ & && (6.14) \end{aligned}$$

Note that we can apply techniques to changes the threshold.

6.8 Excluding a Node from a Cluster

Each node P_j in a cluster will examine its own claim database. If there are enough claims (equal to or greater than the threshold) against a node P_w then the node will broadcast this to all members in the cluster as well as the partial signature of this

claim. In turn, cluster members will respond with the same if they also have greater than or equal to the amount of threshold many claims. Once a threshold amount of claims have been partially signed a full signature can be generated. At this time the cluster can punish the malicious node P_w (dispel them from the group). Immediately the cluster reshares the group signature key. Formally the complaint protocol is given in Algorithm 4.

Algorithm 4 Complaint Protocol

1: **if** Node P_j senses malicious behavior **then**
2: Node P_j stores it inside the private complaint database and broadcasts the complaint and its signature, signed by using $pk_{P_j,Manufacturer}$
3: **else**
4: **if** Node P_i receives complaint about a node **then**
5: Node P_i will store it inside the common complaint database and forward the complaint to other nodes
6: **if** Node P_j has the threshold amount of complaints from the private and common data base **then**
7: Send a signed request of expelling the node from the cluster and a partial signature
8: **if** Node P_i receives threshold amount of requests for expelling a node **then**
9: Node P_i sends a request for expelling the malicious node from cluster with the nodes' signature and ID to all the members for expelling the node
10: Nodes will change their shares or re-key therefore malicious node cannot partially sign messages or hear the cluster members
11: **if** Node P_i claims against node P_j and she put wrong signature on her claim **then**
12: Other nodes will count node p_i as a bad node and they write her in their claim database

6.8.1 Cluster by Cluster Communication

As we are concerned about the safety of a network, cluster by cluster communication needs to be conducted in a secure manner. Nodes in one cluster cannot trust nodes in another cluster and send their data. Clusters need to show some kind of certificate to other neighborhood clusters to make other clusters in their neighborhood trust their information. The cluster certificate contains information about all the cluster members ID, public key, and their manufacturer signature see Fig. 6.5. Because making a public key is expensive, nodes in a cluster can make a symmetric key to communicate with other neighborhood clusters. If cluster C_i wants to communicate with the neighborhood cluster C_j the nearest node in C_i would make a key and she would partially sign it. Then she will send the key to the members in cluster C_i to make a signature on the selected key. Next, she will send the symmetric key which has the group signature to the C_j. After C_j gets the key from cluster C_i, the nodes in cluster C_j also sign their acceptance for using this key for further communication. Therefore, the setup for secure communication between cluster C_i and C_j would be as following:

1. Cluster C_i, sends the cluster certificate and agreed symmetric key to cluster C_j.
2. Cluster C_j checks if the certificate and manufacturer signature are correct.
3. After cluster C_j agreed with the cluster C_i's cluster certificate, cluster C_j, sends the cluster certificate with the acceptance for the symmetric key.
4. Cluster C_i checks if the certificate from cluster C_j is correct or not, if it is right then cluster C_i sends the data or message to cluster C_j.

Other nodes in cluster C_i and C_j know the symmetric key between these two clusters in case of modification other nodes can detect that modification. The cluster certificate might be long and it depends on the number of nodes in a cluster, therefore it might need to use significant bandwidth. Because we want to save energy, we prefer

to have certificate exchange once or repeat it after a long period of time. After the exchange of the cluster certificate, when the cluster C_i wants to send data to the cluster C_j, they would just sign it with cluster signature key. It is one of the reasons why we prefer to renew the shares instead of changing the key in case of malicious behavior in a cluster, because the cluster would need to make another certificate and send it to the neighborhood clusters this and procedure consumes a lot of energy. We should note that symmetric key needs to be changed after some period of time.

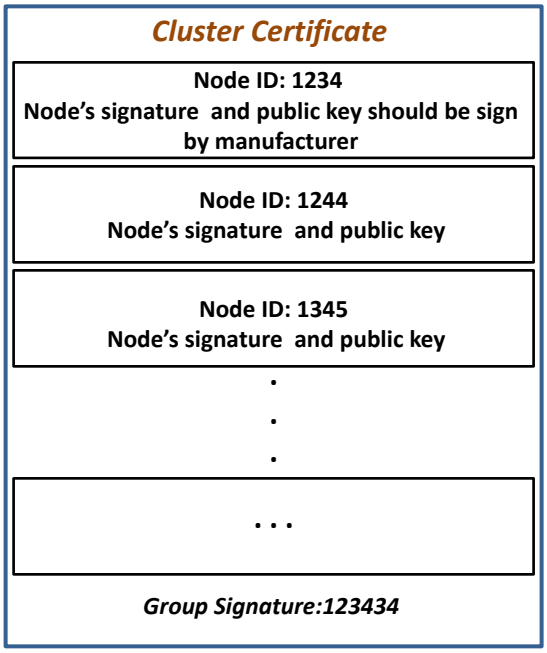

Fig. 6.5.: Cluster Certificate

7. EXPERIMENTAL MODEL AND RESULTS

In our simulation we used a threaded program to simulate the invitation protocol (Algorithm 1). To for control the sequence of nodes, we used a thread called *management* to coordinate the action of the nodes, in the thread programming.A node would pass through several stages as illustrated in Fig. 7.1. S_1, stage 1: nodes are randomly placed in the field (x and y coordinate), S_2, stage 2: nodes flip a coin to decide for being a cluster head or not, S_3, stage 3: if a node is cluster head she will send invitation if not, S_4 stage 4: she will wait for invitation until time T, if she gets an invitation she will accept the first invitation then she will check if her hop number is less than desire hop or not, if it was less then she will send invitation to other nodes in her range. S_5. stage 5: after time T, if a node did not receive any invitations from other nodes, she will become a forced cluster head. To make sure that all the nodes followed all the processes which have been controlled by management, when a nodes wants to enter a process we lock out other nodes' access to that process by using mutex lock [41].

7.1 Discussion

In the [28] Eq. 3.1, the processing center is in the middle of the network field. If we move the base station to the corner of the square field, the result would be:

$$E[D_i|N = n] = \int_A \frac{1}{4a^2}\sqrt{x_i^2 + y_i^2} = 0.765(2a) \tag{7.1}$$

If the total energy used by a sensor to communicate to all inside the cluster where the communication is initialized by the cluster head is e_1, then in our proposal the

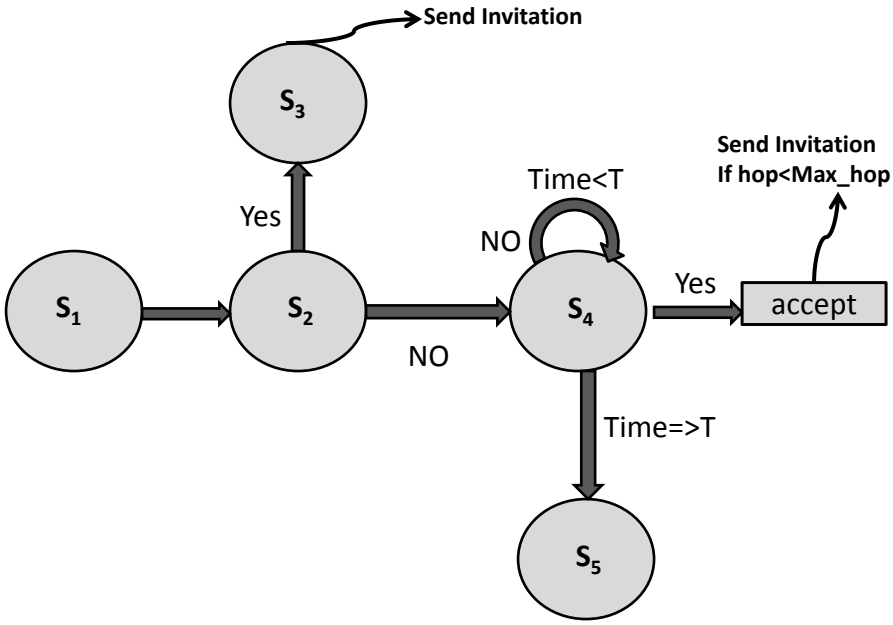

Fig. 7.1.: Invitation Stages

total energy usage for nodes communicating with each other will be less than the Bandyopadhyay and Coyle proposal. In their work if nodes wants to communicate they need to send their data first to the cluster head and cluster head has the responsibility to forward the message to the destination node. In our work nodes find the shortest route to send data, the cluster head would not be central distributer of all the communication,so the route a data would travel is shorter. Therefore, less energy consumption for a network, we would have:

$$e_{opt} \leq \frac{E[L_v|N = n]}{r}. \tag{7.2}$$

Rather, we do not need to calculate the optimal probability p for being a cluster head in a network to save energy. Our greatest concern is securing the data in the network rather than efficiency. Because nodes are monitoring each other, if the number of nodes in a cluster $\mathcal{C}$ is sufficiently large then other clusters can trust $\mathcal{C}$.

7.2 Some Assumptions

If we have N nodes in a field with area of A, the density of network would be:

$$D = N/A. \tag{7.3}$$

If the transmission range for each node in a field is r, and the area that a node cover with her transmission range is $A_T = \pi r^2$. Number of Voronoi cells (clusters) if the nodes perfectly place with **one hop** is:

$$p = \frac{A}{A_T N}, \tag{7.4}$$

in future we represent A_T by $A_{T,1}$ If we have k hops the ideal number of Voronoi cells when nodes perfectly placed in a field would be:

$$A_{T,k} \leq \pi(kr)^2, \tag{7.5}$$

the number of nodes N and area of the field A are free variables, range r, probability p of being a cluster head and hop number would be dependent variables. Range and hop number would affect the number of nodes in a cluster. We observe that the values we would get from these formulas represent ideal values and in practice when nodes are mobile and are not perfectly placed we could not get the same result as we get in the perfect setup for sensors.

For simulating the network, we wanted to fix a security value $\mathcal{V}$, the preferred threshold, which is a minimal number of nodes in a cluster, as mentioned earlier the security of clusters is important and if the number of nodes inside a cluster is enough we would have a suitable amount of security in a cluster. Therefore, parameter $\mathcal{V}$ is important. The probability of a node being a cluster head is $p = \frac{1}{\mathcal{V}}$, if we have N nodes in the whole network, therefore $CL_n = \frac{N}{\mathcal{V}}$, would be the amount of clusters in a network. We can calculate the range from the following:

$$A_{T,k} = \frac{A}{CL_n}. \tag{7.6}$$

In Tables 7.1 and 7.2, we simulated a network with different density values, as one can see if the number of hops increases the number of clusters below security value $\mathcal{V}$ (here less than 7) decreases. In Table 7.1 we wanted to have 7 or more nodes in each cluster (the threshold $\mathcal{V}$ was 7). We could not get a suitable result for such a small range. In Table 7.2 we made some changes on the range by using not restricted threshold, for the $\mathcal{V} = 7$ and $p = \frac{1}{7} = 0.142$ instead of having $CL_n = N/7$ we used $CL_n = N/14$ therefore we could get larger range and the result of that was better. For the case $p = 0.11$, we have $CL_n = N/9$. For the case $p = 0.12$ we have $CL_n = N/8$. As one can see, if we decrease the amount of CL_ns in a network, we achieve better results. Also, from these results we see that if the range and number of hops increase, we decrease the number of clusters, as well as the number of forced cluster heads. All of the results in these two tables are averages, which means we ran calculation 10 times and averaged the results. We did this because the program uses threads, results are dissimilar, due to the latency of the threads running. These results are not predictable, because the nodes randomly will be placed in the network and nodes randomly by flip coins to determine if they will be cluster heads.

In Table 7.3 we show results of increasing the range for the nodes in a network. As illustrated in the table, if the range increases, the average of number of nodes in a cluster would increase, also the number of forced cluster heads decreased. But if you look at Table 7.4, the percentage of the amount of clusters which have less than threshold amount of members increases, which is not what we want for the network. A possible explanation is that we use a mutex to lock other users from the program, when one user find an opportunity, she will send invitation to all of her neighbors. Therefore, if her range is more, more nodes will accept her invitation. While other cluster heads did not have a chance to send invitation to their neighbors. Therefore many of the cluster heads did not get a chance to send an invitation to their neighborhood and these cluster heads would be by themselves. The reason the average number of nodes in the Table 7.3 increases is because some of the clusters have huge amount of nodes inside their cluster. Thus they increase the average. We

ran an experiment to see how many clusters has nodes between 5 to 9, here 7 is the security threshold, as you can see in Table 7.4 if the range increases, the amount of clusters in the range of $5 \leq n \leq 9$ decreases significantly. This means some of the clusters have many members and others not enough members.

In conclusion, nodes need to have a suitable range for sending invitation and for communication. If their range is too small then the number of clusters in a network would increase and also there would be a lot of forced cluster heads. If the range is too large, in addition to the cost of energy, we would have more cluster heads who do not receive acceptances to their invitations, so there would be less clusters with the desired amount of nodes in a cluster.

In order to calculate the cost of sending an invitation, we calculated the cost of a broadcast by finding the cluster (tree) leaves L_c. Therefore, if we have n_c nodes in a group with L_c leaves, see Fig. 7.2, the broadcast cost in a group is:

$$cost_c = n_c - L_c. \tag{7.7}$$

In Fig. 7.2, there are 10 nodes, 5 of the nodes are tree leaves. Therefore, the cost of a broadcast to send an invitation in this tree (cluster) is 5. So the cost of broadcast for invitation for the whole cluster, where the number of clusters in a network is, CL_n would be:

$$Cost_{broadcast} = \sum_{i=0}^{CL_n} (n_c - L_c). \tag{7.8}$$

The average cost would be:

$$Avg_{Cost} = \frac{Cost_{broadcast}}{CL_n}. \tag{7.9}$$

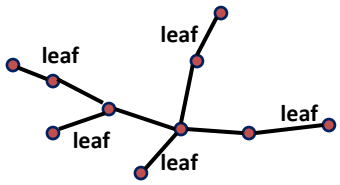

Fig. 7.2.: Broadcast Tree

Table 7.1: Restricted Range

d	N	r	hop	prob p	CL_n	$n = 1$	$n \leq Th$	n_{Avg}	Forced CH
6	350	0.48	3	0.142	105.3	61.4	83.7	3.35	53.9
6	350	0.48	2	0.142	110.8	66.4	91.4	3.23	59.4
6	350	0.48	1	0.142	175.7	131.6	164.3	2	124.3

Table 7.2: Non-restricted Range Results

d	N	r	hop	prob p	CL_n	$n = 1$	$n \leq Th$	n_{Avg}	Forced CH
5	350	0.56	3	0.142	61	21.7	37.4	5.79	9.6
5	350	0.56	2	0.142	65.1	24.3	42.4	5.4	13.7
5	350	0.56	1	0.142	106.3	65.7	85.6	3.31	54.9
6	350	0.67	3	0.142	59	17.4	35.6	5.97	7.6
6	350	0.67	2	0.142	65.2	23.8	42.4	5.41	13.8
6	350	0.67	1	0.142	101.6	60.1	80.7	3.49	50.2
7	350	0.79	3	0.142	63	22.6	39.5	5.58	11.6
7	350	0.79	2	0.142	64.7	23.9	40.7	5.51	13.3
7	350	0.79	1	0.142	106.3	64.3	86.6	3.33	54.9
8	350	0.9	3	0.142	60.9	21.2	37.3	5.82	9.5
8	350	0.9	2	0.142	68.5	28.4	45.8	5.23	17.1
8	350	0.9	1	0.142	102.1	61.5	80.7	3.4	50.7
8	300	0.97	3	0.142	52	18.1	32.3	5.7	9.9
8	300	0.97	2	0.142	62.2	30.1	43.5	4.9	20.1
8	300	0.97	1	0.142	99.7	66.3	82	3.1	57.6
6	350	0.67	2	0.11	67	32.4	49.9	5.4	26.8
6	350	0.67	2	0.12	62.3	25.1	42	5.7	17

Table 7.3: Different Range

	r	CL_n	$n = 1$	$n < Th$	n_{Avg}	Forced CH	A_T
1	0.97	87.4	46.8	66.1	4.03	36	2.95
2	1.07	72.3	33.9	51.5	4.9	20.9	3.59
3	1.17	67.8	32.2	47.8	5.21	16.4	4.3
4	1.27	62.1	29.7	43	5.69	10.7	5.06
5	1.37	57.4	27	39.1	6.18	6	5.86
6	11.47	56.3	29.7	40.3	6.27	4.9	6.78
7	1.57	58.8	33.4	43.5	6.04	7.4	7.74
8	1.67	52.4	29.2	38.2	6.74	1	8.76
9	1.77	53.1	32.1	39.6	6.65	1.7	9.84
10	1.87	52.1	32.3	40.1	6.77	0.7	10.98
11	1.97	52	32.7	38.9	6.78	0.6	12.19
12	2.07	51.8	35.3	40.4	6.86	0.4	13.46
13	2.17	51.4	35.2	40.4	6.86	0	14.7
14	2.27	51.5	35.5	40.4	6.85	0.1	16.1

Table 7.4: Less Than Threshold

row	$n \geq Th$	percent	$5 < n < 9$
1	21.3	24.3	12.5
4	19.1	30.7	7.7
8	14.2	27	4.2
11	13.1	25.1	3.4
14	11.1	21.5	3.3

8. FUTURE WORK AND CONCLUSION

In this work, we have constructed mechanism that secure a sensor network and achieve it in an energy efficient manner. Essential to this construction is the ability to monitor the network in small groups/clusters. In order to achieve this we have constructed an efficient invitation protocols. The goal is to provide a sufficiently high percentage of clusters whose membership exceed a suitable security threshold $\mathcal{V}$. The invitation protocol is an important contribution We assumed that all the nodes have the same range and they do not change their range.

For future work one should consider the situation that nodes could utilize different reception range and in the cases they did not receive any invitation, they may increase their reception range after time T. Then we would have less clusters, also more clusters would satisfy the the security threshold $\mathcal{V}$. But this would come at a energy cost and such a trade-off would need to be carefully analyzed.

There is a problem with new nodes, if new nodes are generated and join the network, there is a problem that a cluster cannot determine if the nodes are actually new nodes. The node which is malicious may masquerade a new node. The nodes manufactured date does alleviate this problem to some degree.

There is another problem, there is a possibility that a malicious node may keep joining and leaving clusters before she gets any bad reviews she leaves a group. One needs to find mechanisms that identifies and prevents this behavior.

LIST OF REFERENCES

LIST OF REFERENCES

[1] "Sensor fault found in cork plane crash inquiry.", *http://blog.zacharyabel.com/category/computer-science/*. Last date accessed June 30, 2014.

[2] J. V. D. Merwe, D. Dawoud, and S. McDonald, "A survey on peer-to-peer key management for mobile ad hoc networks," *ACM computing surveys (CSUR)*, vol. 39, no. 1, p. 1, 2007.

[3] R. Rajagopalan and P. Varshney, "Data-aggregation Techniques in Sensor Networks: A Survey," *IEEE Commun. Surveys Tutorials*, vol. 8, no. 4, pp. 4863, Oct. 2006.

[4] D. Wei and H. A. Chan, "Clustering ad hoc networks: Schemes and classifications," in *Sensor and Ad Hoc Communications and Networks, 2006. SECON'06. 2006 3rd Annual IEEE Communications Society*, vol. 3, pp. 920–926, IEEE, 2006.

[5] Q. Li, J. Aslam, and D. Rus, "Hierarchical power-aware routing in sensor networks," in *Proceedings of the DIMACS workshop on pervasive networking*, Citeseer, 2001.

[6] T. Issariyakul and E. Hossain, *Introduction to network simulator NS2*. Springer, 2011.

[7] D. R. Stinson, *Cryptography: theory and practice*, vol. 36. CRC press, 2006.

[8] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.

[9] W. Trappe and C. Lawrence, "Washington. 2006," *Introduction to Cryptography with Coding Theory*.

[10] J. Pieprzyk, T. Hardjono, and J. Seberry, *Fundamentals of computer security*. Springer, 2003.

[11] C.-M. Li, T. Hwang, and N.-Y. Lee, "Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders," in *Advances in CryptologyEUROCRYPT'94*, pp. 194–204, Springer, 1995.

[12] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," in *Advances in CryptologyEUROCRYPT'94*, pp. 275–286, Springer, 1995.

[13] S. Ghiasi, A. Srivastava, X. Yang, and M. Sarrafzadeh, "Optimal energy aware clustering in sensor networks," *Sensors*, vol. 2, no. 7, pp. 258–269, 2002.

[14] V. Kawadia and P. Kumar, "Power control and clustering in ad hoc networks," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 1, pp. 459–469, IEEE, 2003.

[15] Y. Fernandess and D. Malkhi, "K-clustering in wireless ad hoc networks," in *Proceedings of the second ACM international workshop on Principles of mobile computing*, pp. 31–37, ACM, 2002.

[16] S. Srivastava and R. Ghosh, "Cluster based routing using a k-tree core backbone for mobile ad hoc networks," in *Proceedings of the 6th international workshop on Discrete algorithms and methods for mobile computing and communications*, pp. 14–23, ACM, 2002.

[17] S. Banerjee and S. Khuller, "A clustering scheme for hierarchical control in multi-hop wireless networks," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, pp. 1028–1037, IEEE, 2001.

[18] S. Sivavakeesar, G. Pavlou, C. Bohoris, and A. Liotta, "Effective management through prediction-based clustering approach in the next-generation ad hoc networks," in *Communications, 2004 IEEE International Conference*, vol. 7, pp. 4326–4330 Vol.7, June 2004.

[19] B. An and S. Papavassiliou, "A mobility-based clustering approach to support mobility management and multicast routing in mobile ad-hoc wireless networks," *International Journal of Network Management*, vol. 11, no. 6, pp. 387–395, 2001.

[20] M. K. Denko, "The use of mobile agents for clustering in mobile ad hoc networks," in *Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology*, pp. 241–247, South African Institute for Computer Scientists and Information Technologists, 2003.

[21] P. Rabinovich and R. Simon, "Secure aggregation in sensor networks using neighborhood watch," in *Communications, 2007. ICC'07. IEEE International Conference on*, pp. 1484–1491, IEEE, 2007.

[22] B. Zhu, S. Setia, and S. Jajodia, "Providing witness anonymity in peer-to-peer systems," in *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 6–16, ACM, 2006.

[23] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 255–265, ACM, 2000.

[24] S. Buchegger and J.-Y. Le Boudee, "Self-policing mobile ad hoc networks by reputation systems," *Communications Magazine, IEEE*, vol. 43, pp. 101–107, July 2005.

[25] J.-S. Liu and C.-H. Richard Lin, "Energy-efficiency clustering protocol in wireless sensor networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 371–388, 2005.

[26] G. Pei and C. Chien, "Low power tdma in large wireless sensor networks," in *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, vol. 1, pp. 347–351, IEEE, 2001.

[27] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[28] S. Bandyopadhyay and E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3, pp. 1713–1723, IEEE, 2003.

[29] S. Foss and S. Zuyev, "On a voronoi aggregative process related to a bivariate poisson process," *Advances in Applied Probability*, pp. 965–981, 1996.

[30] S. ZareAfifi, R. Verma, B. King, P. Salama, and D. Kim, "Secure countermeasures to data aggregation attacks on sensor networks," in *Circuits and Systems (MWSCAS), 2012 IEEE 55th International Midwest Symposium*, pp. 856–859, IEEE, 2012.

[31] N. I. Al-Najjar, "Aggregation and the law of large numbers in large economies," *Games and Economic Behavior*, vol. 47, no. 1, pp. 1–35, 2004.

[32] C. E. Leiserson, R. L. Rivest, C. Stein, and T. H. Cormen, *Introduction to algorithms*. MIT press, 2001.

[33] D. Wagner, "Resilient aggregation in sensor networks," in *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pp. 78–87, ACM, 2004.

[34] S. Ganeriwal, L. K. Balzano, and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 4, no. 3, p. 15, 2008.

[35] "Voroni cell." *http://www.cse.ohio-state.edu/~tamaldey/course/784/vordel.pdf*. Last date accessed June 30, 2014. *http://blog.zacharyabel.com/category/computer-science/*, Last visited on June 30, 2014.

[36] Y. Desmedt, "Society and group oriented cryptography: A new concept," in *Advances in CryptologyCrypto87*, pp. 120–127, Springer, 1988.

[37] T. P. Pedersen, "A threshold cryptosystem without a trusted party," in *Advances in CryptologyEUROCRYPT91*, pp. 522–526, Springer, 1991.

[38] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," in *Advances in CryptologyCRYPTO89 Proceedings*, pp. 307–315, Springer, 1990.

[39] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive secret sharing or: How to cope with perpetual leakage," in *Advances in CryptologyCRYPT095*, pp. 339–352, Springer, 1995.

[40] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[41] A. Silberschatz, P. B. Galvin, G. Gagne, and A. Silberschatz, *Operating system concepts*, vol. 4. Addison-Wesley Reading, 1998.