Secure Data Aggregation Protocol with Byzantine Robustness for Wireless Sensor Networks

by

Tarek Khalifa

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Master of Applied Science

in

Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2007

Abstract

Sensor networks are dense wireless networks constituting of small and low-cost sensors that collect and disseminate sensory data. They have gained great attention in recent years due to their ability to offer economical and effective solutions in a variety of fields; and their profound suitability to address mission critical problems that are common in health, transportation, and military applications. “Sensor networks” is a technology that is seen to change the world, and as such their deployment is expected to see a rapid growth.

Effective security strategy is essential for any sensor network in order to maintain trustful and reliable functionality, protect sensory information, and ensure network component authenticity. Security models and protocols that are typically used in other types of networks, such as wired networks, are not suitable for sensor networks due to their specific hardware specifications.

This thesis highlights some of the research done so far in the area of security of wireless sensor networks and proposes a solution to detect Byzantine behaviour - a challenging security threat that many sensor networks face. The proposed solution’s use of cryptography is kept at a minimum to ensure maximum secure bandwidth.

Under this solution, a sensor network continues to work normally until an attack is suspected. Once an attack is suspected, a cryptography scheme is enabled to authenticate suspected nodes and to allow the identification of potential external attacks. If an attack seems to persist after the cryptography scheme has been enabled, the same mechanism is used to identify and isolate potentially compromised nodes.

The goal is to introduce a degree of intelligence into such networks and consequently improve reliability of data collection, accuracy of aggregated data, and prolong network lifetime.

Acknowledgments

My thanks and praise first and foremost go to Almighty God for giving me the knowledge, opportunity, and the strength to accomplish this work.

Along the way, several people deserve sincere recognition. I would like first to extend my sincere thanks and gratitude to my advisor, Dr. Otman Basir, for his valuable advice and guidance and for his generous help and time. His constant support was the secret of achieving all this work. I would like also to extend my appreciation and thanks to my thesis committee members, Dr. Pin-Han Ho and Dr. Xuemin (Sherman) Shen , for sparing their time in reviewing this thesis report.

Thanks go to my friends for their prayers and support in the completion of this work. I am very fortunate to have so many exceptional and genuine people in my life.

I would like to thank all my family members. My parents, Dr. Masaud Khalifa and Jumia Ali, and my brothers and sisters, Dr. Nagib, Dr. Manal, Dr. Najwa, Marwa, Suhaib, Faisal, and Gassan, have been the greatest supporters in my life. I am grateful to them for their unconditional love, support, encouragement and sacrifices they made to raise me.

Last but not least, a heartful gratitude goes to my beloved wife, Aziza, for the endless understanding, support, patience and care show showed during this important period in my career. Despite her busy school work and being thousands of miles away from her family, she has always been available to uplift me when I felt down and to congratulate me when I achieved. I am by all means indebt to her.

This thesis is dedicated to mum and dad

I love you both

Table of Contents

List of Tables

List of Figures

CHAPTER 1 INTRODUCTION

"Sensor networks" is a term used to refer to a heterogeneous system combining tiny sensors and actuators with general/special-purpose processors. Sensor networks are assumed to grow in size to include hundreds or thousands of low-power, low-cost, static or mobile nodes.

Sensor networks are useful in a variety of fields, including environmental monitoring, military surveillance, and information gathering from inhospitable places. They not only monitor but also facilitate control of physical environments from remote locations. Sensors play important roles in various applications: measuring flow, temperature, humidity, pressure, brightness, mechanical stress, and proximity. Areas such as disaster anticipation, environment control, health care, military command control benefit greatly from this emerging technology. The Media has made important statements in favor of wireless sensor technology. In 2003, MIT's Technology Review magazine described sensor networks as "one of the ten technologies that will change the world." Moreover, ON World, a wireless research firm [1], stated in a report that more than half a billion nodes will be available for wireless sensor applications by 2010 for an end-user market worth at least $7 billion.

In many sensor network applications, such as military applications, data integrity, and quite often confidentiality, are high priorities, leading to the question of *to what degree the network is secure*? So far, most of the research has focused on making sensor networks a reality. Security, relatively speaking, has not received as much concern primarily because of the difficulty of dealing with such devices under stringent specifications (e.g., cost, computation power, memory space, energy) Applying conventional security techniques (cryptography) in such networks is impractical since these techniques require high processing capability. Traditionally, security relies heavily on cryptographic methods; nevertheless, a significant number of problems require security specification that are beyond the scope and ability of all known cryptographic techniques; for example, cryptographic methods are blind to the quality of sensory data and it is quite possible that the security cryptic data is already compromised.

Recalling our experience with the Internet and cellular networks, one can say that if the security of a given network architecture is not properly addressed from the very beginning, its security can easily be breached by malicious attacks. For instance, most of the existing routing protocols have been designed without security as a goal. Consequently, all of them are highly susceptible to all sorts of attacks. An attacker can divert some traffic, increase latency, or even bring the entire network down sometimes with as little effort as sending a single packet.

However, introducing or reinforcing security mechanisms posteriori can be an expensive process and often can be ineffective. This principle equally applies to sensor networks.

Wagner et. al., in [2], show a number of examples in which simple attacks were able to bring down a network running some known routing protocols, for example, TinyOS beaconing protocol. This protocol constructs a breadth first spanning tree rooted at a base station. A route update is initiated at the root and broadcast to the neighboring nodes, which, in turn, propagate the same update to the other nodes. Each node marks the sender as a parent node (Figure 1-1).

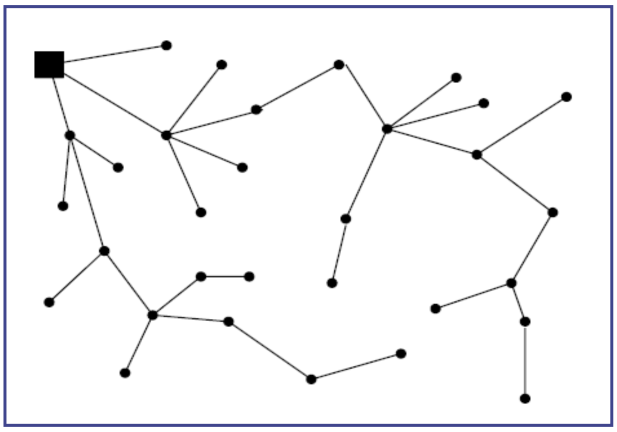


Figure 1-1 A WSN Constructed Using TinyOS

They have discussed a number of attacks, which are highlighted below.

1. Spoofing Routing Information: Since routing updates are not authenticated, any node can claim to be a base station and, as a result, attract all the traffic (Figure 1-2).

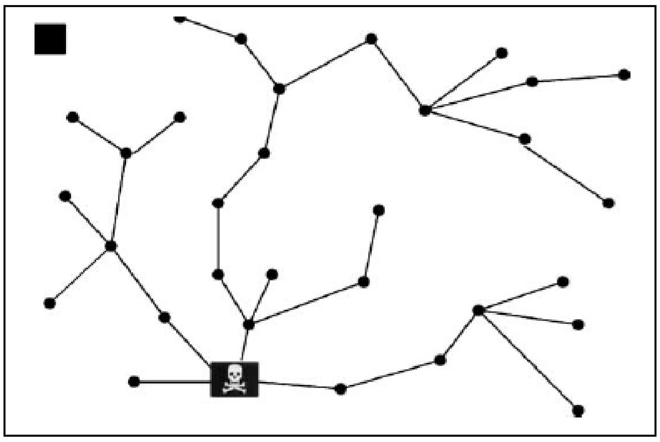


Figure 1-2 An Adversary Spoofing a Routing Update from a Base Station

2. Wormhole Attack: Implementing an authentication mechanism does not completely secure the protocol. An adversary who is interested in eavesdropping on, selectively modifying, or dropping off packets can still do so by mounting two colluding laptop-class nodes, one near the base station and one near the targeted area. The first node forwards authenticated routing updates to the second node

through an out-of-band communication channel in just a single hop, which makes it faster that it would be normally through multi-hop routing. They will create two isolated subtrees, with the second in the targeted area. Because the far subtree receives the routing update through the second adversary node, all the nodes will mark it as the parent node, resulting in all the traffic being tunnelled through it (Figure 1-3).

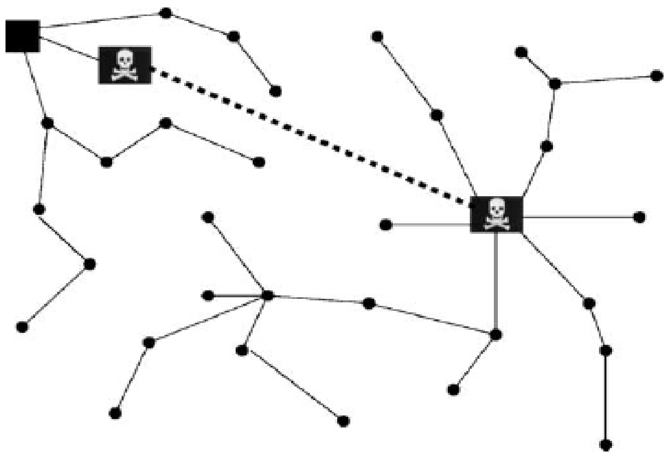


Figure 1-3 Canalizing All Traffic through Two Colluding Nodes

3. Hello Flood Attack: Another simple attack, but an effective one, that can cripple the network is where a powerful adversary node can broadcast a HELLO packet to the entire network. All the nodes that receive the packet will mark the adversary as their parent even if their transmission range does not allow them to communicate back to the adversary (Figure 1-4).

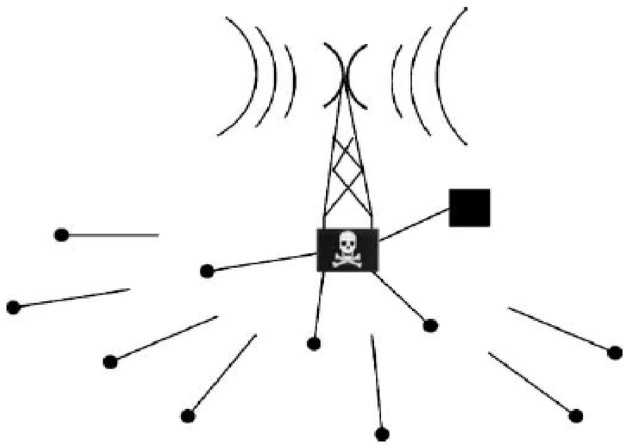


Figure 1-4 HELLO Flood Attack

4. Loops: Routing loops can easily be created. An adversary can learn about a node's neighbours and forge routing information propagated in the network. For example, it can let two nodes mark each other as parents by spoofing source addresses, causing all the traffic between the two nodes to bounce back and forth whenever messages are sent to either one of them.

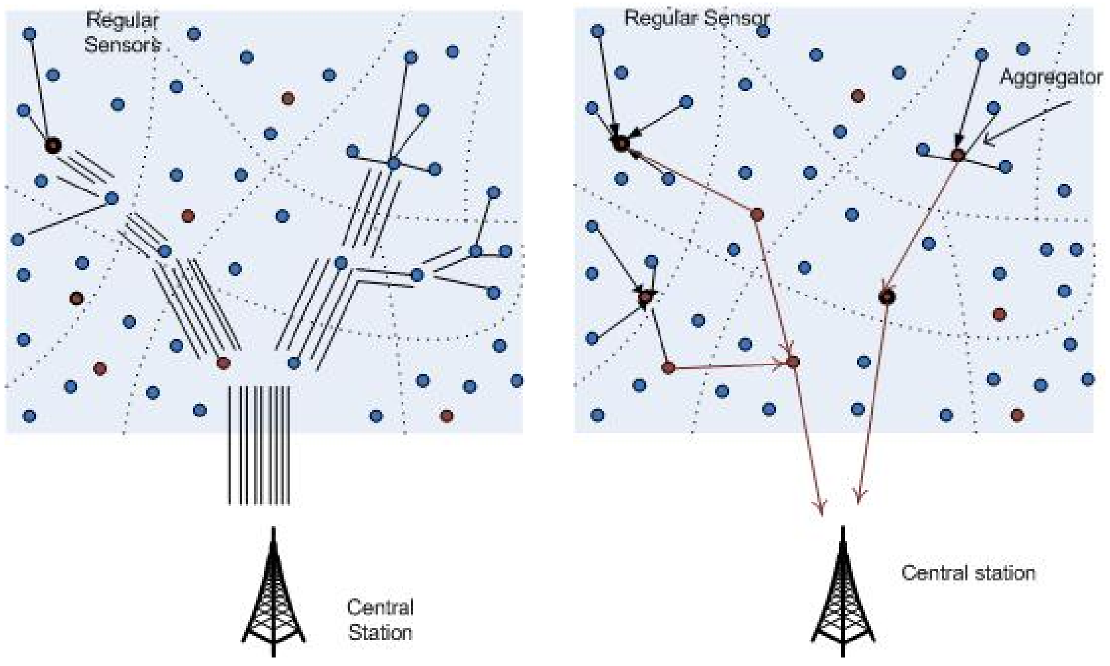
Security may become more demanding in certain applications than in others, e.g., military related services; yet it cannot be eliminated from any applications. Let us see a regulating application in which, for instance, a sensor network is installed in a large building to measure temperature. Measurements are sent to a control station in which the temperature average is calculated and, accordingly, a decision is made. If the average temperature exceeds a certain threshold, then the air conditioner is turned on. However, if there is not enough security in place, then the system will be prone to failure. If somebody knows where the sensors are, then he/she can simply overheat a single sensor and shift the average temperature, which may cause the air conditioner to turn on when it should not.

The previous examples stress the vitality of embedding security in routing protocols for the acceptance and use of sensor networks in many applications. Many of the existing routing protocols have the same deficiency. Wagner et al. provide a good analysis of some of the existing routing protocols and show their respective vulnerabilities. For example, Directed Diffusion, Geographic Routing, Minimum Cost Forwarding, LEACH, and Rumor Routing are considered highly susceptible to similar failures.

Although sensor networks can be seen as a special type of ad hoc network, the existing network protocols that are designed for ad hoc networks, including the security protocols, cannot be implemented on sensor networks due to the following reasons are related:

- Sensor nodes tend to be small, inexpensive, and very limited in hardware resources (processing power, storage, bandwidth, and energy),
- Sensor nodes are very prone to failure,
- Sensor nodes lack physical security, and
- Sensor nodes are deployed densely in a large number because of their short range communication ability.

Newly designed protocols, therefore, are required to suit the special needs of the sensor networks. For example, in-network processing, such as data aggregation, is indubitably required to cut down the amount of data transmission. Summarizing data, which is usually correlated, while travelling in the network towards the final collection point, helps improve the utilization of resources such as bandwidth and battery power. The fact that approximately 70% of the total energy is consumed on communication leads to the conclusion that data aggregation has the potential to tremendously enhance the network lifetime. Figure 1-5 depicts the difference between a network without data aggregation and another with data aggregation.



(a) Without Data Aggregation (b) With Data Aggregation

Figure 1-5 Forwarding Packets With and Without Data Aggregation

Data Aggregation and Security Challenges

With the importance of in-network processing, however, enforcing security becomes a more challenging task. As a matter of fact, data aggregation techniques and security protocols face conflicts in their implementation. On one side, to eliminate redundancy of data and thereby reduce the number of packets transmitted in the network, the data aggregation protocols require sensor data to be processed by the intermediate nodes as much as possible. Therefore, data should be available in the clear text at every intermediate node to perform the aggregation process. On the other side, security protocols commonly require that sensor nodes encrypt any data prior to transmission so that information confidentiality is achieved. Moreover, it is desired that data be encrypted at the source node, the sensing sensor, and stay encrypted all the way until it reaches the final destination, the home server, so that end-to-end security is realized. Handling data this way prevents information from leaking to a third party even if some of the intermediate nodes are malicious. The conflict occurs if the security protocol prevents decryption of data at the intermediate nodes; data aggregation then cannot be applied. Nevertheless, if intermediate nodes are given the ability to decrypt received sensor data, a compromised node can simply lie about the aggregation results, rendering the network useless. As a result, security protocols must handle this conflict by allowing data aggregation to take place while data integrity is maintained.

The question that arises is how to meet those demands, especially with the existence of such new attacks as the node compromise attacks? In this newly emerging type of attack, the enemy gains full control over sensor nodes and aims at injecting false data into the network. Injecting false data can happen either through compromised or malfunctioning nodes. Consequences include violated data integrity requirement, false alarms, depleted battery power, and weakened computational and communication resources. In fact, the compromised nodes are able to defeat security cryptographic systems because they already possess the secret keys and, therefore, can successfully authenticate the false data to a neighboring node. Nonetheless, data aggregation cannot be sacrificed. Its high importance in reducing redundancy, expanding network lifetime, and enhancing data accuracy necessitates its implementation. However, both data aggregation and false data infection cause sensor data modification, and so legitimate data and false data can be confused. For those reasons, false data detection, compromised node elimination, and data aggregation protocols should be designed together so that the sensor network can survive and work successfully.

This thesis addresses the problem of Byzantine faults, i.e., malicious nodes as well as malfunctioning ones. Those kinds of nodes, by definition, may report bad measurements deliberately, as in the case of malicious behaviour, or not deliberately, as in the case of malfunction. Our objective is to secure wireless sensor networks that are based on collecting data from a group of sensors and aggregating the fused data at special nodes, called aggregators, which, in turn, deliver the results to a central station. Our scheme ensures the validity of data and evaluates the honesty of sensor nodes.

Presumably, cryptography promises a high level of security. This is quite true. Nevertheless, due to the poor HW capabilities of sensors, applying cryptographic tools would be very costly. Thus, cryptography tools are used only when needed; otherwise, communication between nodes continues without such tools. In other words, if a node is suspected to be behaving maliciously (lying about its measurement), that node is forced to use its stored keys for future communication, which eliminates possible outside attacks. If the misbehaviour continues, then the node is filtered out of the system, a possible inside attack being assumed. We use Generalized Extreme Studentized Algorithm (GESD) to detect Byzantine sensors, combined with an information theoretic measure to qualify trustworthiness of each sensor and to decide when a sensor is to be considered Byzantine. With this scheme, the network becomes intelligent, and energy is preserved; thus, its lifetime is increased.

The rest of this thesis is organized as follows. Chapter 2 provides background information on wireless sensor networks, their nature, applications, and typical paradigms. Their vulnerabilities and the challenges they face for employing security are also discussed in this chapter. Chapter 3 documents the two main concepts, security in sensor networks, and data aggregation techniques in sensor networks, with an extensive list of the work done in these two directions. Moreover, the concept of Byzantine behaviour is introduced here to clarify the Byzantine type of threat to sensor networks. Next, Chapter 4 provides the details of the proposed approach to achieving secure data aggregation with the existence of Byzantine nodes. Simulation results are presented in this chapter. Finally, concluding remarks and future research directions appear in Chapter 5.

CHAPTER 2 BACKGROUND: WIRELESS SENSOR NETWORKS

As defined earlier, wireless sensor networks are wireless networks that consist of individual nodes that are able to measure and communicate certain environmental phenomena using sensors; the nodes of a sensor network collaborate among one another to deliver measurements in the form of messages over wireless communication channels to a central station; without collaboration, a single node cannot fulfil the desired goal [3].

This chapter investigates typical software and hardware specifications of such networks and discusses their protocol stacks, applications, deployment paradigms. The chapter highlights some of the challenges of employing security mechanisms in sensor networks.

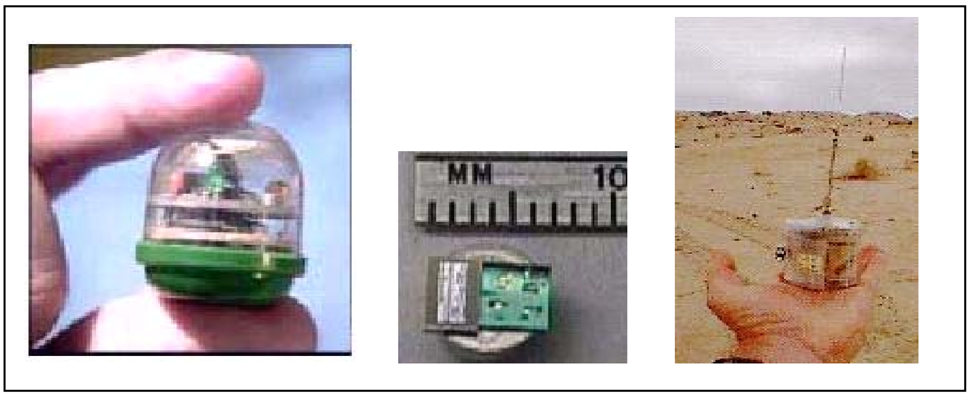


Figure 2-1 Examples of sensor nodes

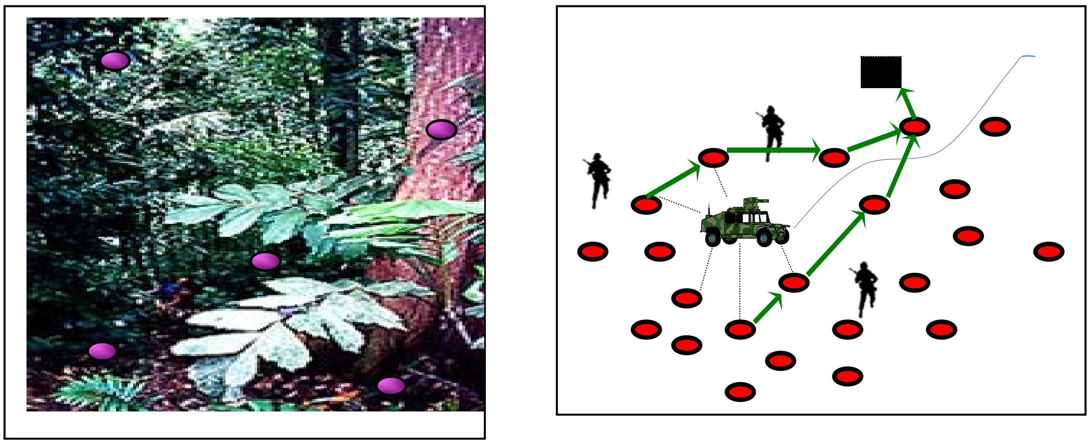
2.1 Sensor Networks

Sensor networks rely on sensing, processing and wireless communication abilities. Thanks to recent enhancements and developments in electronics, sensor networks have greater flexibility in terms of the solutions they can offer in a wide range of applications. It is now possible to implement more effective ways to trace or learn about certain physical phenomena in different environments, friendly or hostile. Nevertheless, sensor networks face some challenges that constrain or make difficult their implementation in certain areas. Those challenges are addressed in this chapter.

It is envisioned that sensor networks will eventually cover and instrument the entire planet. They will continuously monitor and gather information on diverse phenomena, including endangered species, soil and air contaminants, patients, and industrial environments. However, their extent application is only limited by the availability of the sensing elements that can be employed. Some of the sensors used today include those that measure temperature, pressure, humidity, flow, vibration, brightness, mechanical stress, and proximity. Thus, sensor networks are well suited to a variety of monitoring and surveillance applications [4], including (Figure 2-2):

- Wildlife monitoring.
- Bushfire response.
- Military command.
- Distributed robotics.
- Industrial quality control.
- Observation of critical infrastructure.
- Smart buildings.
- Traffic monitoring.

- Examining human heart.
- Precision agriculture.



(a) Environmental Monitoring (b) Battle Field

Figure 2-2 Examples of Sensor Network Applications

2.2 Sensor Hardware Considerations

The development of sensor nodes (hardware and software) has been greatly influenced by the type of application they serve. Generally, sensor nodes must be small, economical, energy efficient, equipped with sensing elements, good at computation performance, and have suitable wireless communication facilities. Figure 2-3 shows the main hardware components that build a typical sensor node: processor, memory, sensors, communication elements, and power supply [5,6,7]. However, it is important to note that some applications may require extra hardware components, for example, a GPS to locate a node, or UAVs to move a node, or a power generator. There follows an explanation of each of the four main components mentioned above:

1. Processor: consists of a processing unit, memory, and peripheral modules such as ADC, UART, timers, and interfaces connected through buses. The processor takes on the role of executing the code, such as the operating system and the application software, and also deals with data (acquisition, preprocessing and processing of incoming and outgoing information).
2. Memory: similar to any computerized system, RAM and EPROM are used. The former is used for storing data samples under processing or messages either received or being sent. The latter (EEPROMs or flash EPROMs) are used to store the program code.
3. Sensing Unit: consists of a sensor and ADC. It facilitates the interface unit to the environment. A variety of sensors are available today for sensing different physical phenomena. In terms of operation mechanism, sensors can be classified in three categories:
    - Passive omni-directional: sensors of this type of sensors measure a physical quantity without having to interfere with the environment, as, for instance, those that measure temperature, light, humidity and pressure.
    - Passive, narrow-beam: sensors of this type have to take a certain direction. Examples include ultrasonic sensors and cameras.
    - Active: such sensors provide measurements for samples collected from the environment. Radars and seismic sensors are of this type.

4. Communication Elements: called transceivers; have the capability of transmission and reception at the same time. Their fundamental task is to

convert bit streams coming internally from the controller into waves and vice versa. It is this unit that maintains the connection among nodes and forms the network. Radio frequency (RF) is commonly used because it offers a long range of communication at high rates and has a fairly small error rate. Besides, it does not require direct sight between transmitter and receiver. Other wireless communication options, such as optical, ultrasound, and electromagnetic, are also utilized in certain applications.

5. Power Unit: an important component because it determines the lifetime of a node, hence the entire network. Usually a node feeds on a temporary source of power such as batteries. Other options include scavenging energy from the environment, e.g., using solar cells.

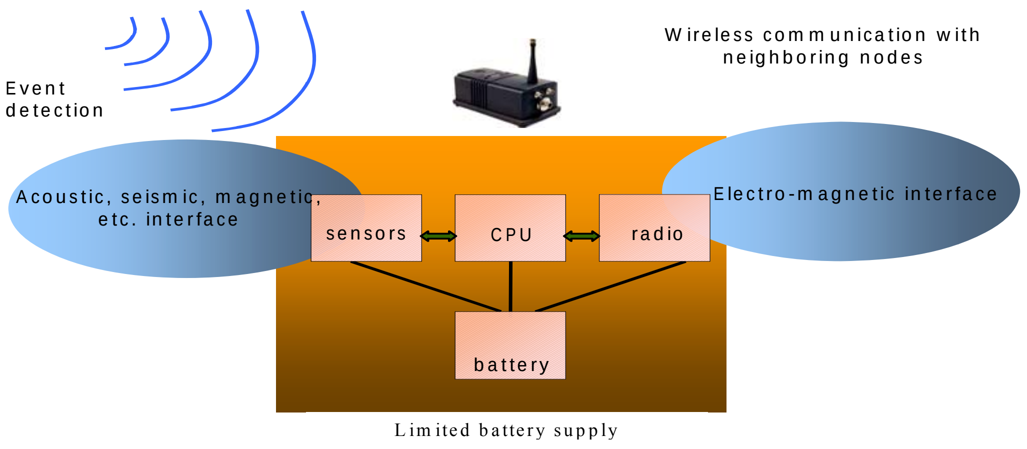

Event detection
Wireless communication with neighboring nodes
Acoustic, seismic, magnetic, etc. interface
sensors
CPU
radio
Electro-magnetic interface
battery
Limited battery supply

Figure 2-3: Sensor Node Hardware Components

Today, Companies such as Crossbow [www.xbow.com], Intel [www.intel.com], and Ambient Systems [www.ambientsystems.com], develop, manufacture, and sell sensor nodes. Table 1 lists hardware specifications for three commonly used sensor nodes in research and commerce.

Table 1: Sensor HW Specifications

	MICAZ	µNODE	Intel mote
CPU	4MHz, 8-bit Atmel	16MHz, 16bit	16-bit, 12 MHz
Flash memory	512KB	48KB	512 KB
RAM	4KB	10KB	64 KB
Radio frequency	916MHz, 30m range	868/917MHz, 50m	900 MHz
Bandwidth	40Kbps	50kbps	100 Kbps

Improvement in the hardware specifications is on going. MICA2, the next generation of motes by Crossbow, for example, is designed with a 32 bit processor, 32MB RAM, and 32MB Flash memory, but this mote is still very expensive.

2.3 Sensor Node Communication Architecture (Protocol Stack)

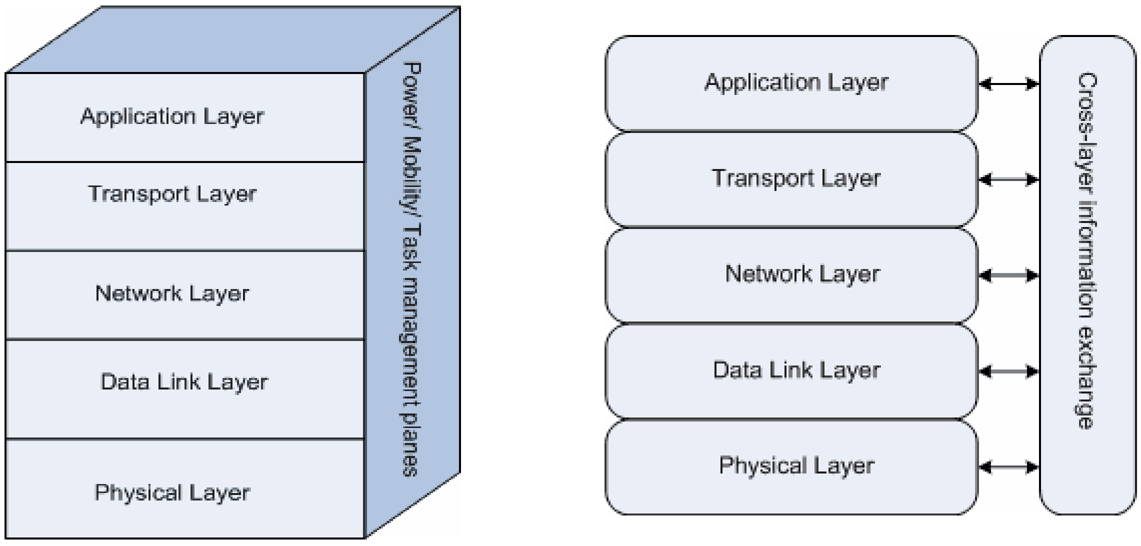
Similar to all other communication devices, sensor network design complies with the layer design approach, in which every layer has to provide well-defined functionalities. According to [8], the protocol stack consists of the physical layer, data link layer, network layer, transport layer, and application layer (Figure 2-4), which are discussed below:

- *The Physical Layer*: handles transmission, modulation, and receiving bit streams between two nodes. Frequency selection, carrier frequency generation, signal detection, modulation, and data encryption take place at this layer. In the design of the physical layer, energy becomes the most important aspect among many other issues, such as signal decay, scattering, shadowing, reflection, diffraction, multi-path, and fading.

Techniques such as ultrawideband (UWB) and impulse radio (IR) have been used to conserve energy.

- *The Data Link Layer*: handles medium access, error control, multiplexing of data streams, and data frame detection. It ensures reliable point-to-point and broadcast connections. MAC protocol, as part of the data link layer, has to be energy aware and has to minimize collisions. Some examples of MAC protocols that are tailored specifically for sensor networks are Self-organized Medium Access Control (SMAC) [9], Eavesdrop and Register (EARS) [9], and Hybrid TDMA/FDMA-based protocols [10].
- *The Network Layer*: deals specifically with packet routing, i.e., finding the most efficient path for a packet to go from a node to a given destination. Sensors form different paradigms for different applications and network differently from all other types of networks; hence, routing protocols have to be designed specifically for the sensor network. Relatively, the network layer has received the most attention, resulting in a variety of routing protocols, which can be categorized in the following techniques: flooding, gossiping, Sensor Protocols for Information via Negotiation (SPIN) [11], SAR (Sequential Assignment Routing) [9], Directed Diffusion [12], and many other routing techniques.
- *The Transport Layer*: handles the flow of packets between two multi-hop-away nodes. Nevertheless, the transport layer is not required in most sensor network applications. So far, research has not gone so deep in developing this layer as to match the specific requirements of sensor networks. One anticipated need is when connection to the Internet is required.

- *The Application Layer*: represents the application software that handles certain tasks directly, which can be different from one application to another. With such a variety of application areas in sensor networks, protocols at this layer have to be flexible in order to provide transparent access to lower hardware and software layers. Research has yet to be conducted to develop new protocols and to enhance the current protocols, such as Sensor Management Protocol (SMP), Task Assignment and Data Advertisement Protocol (TADAP), and Sensor Query and Data Dissemination Protocol (SQDDP) (the three protocols are discussed in [13]).



(a) Protocol Stack (b) Cross-layer information exchange

Figure 2-4: Sensor Network Communication Architecture

As Figure 2-4(a) shows, there are three additional layers in another dimension: power management plane, mobility management plane, and task management plane. The first one manages how a sensor consumes its power economically, for example, turn off the sensor when it is idle. The second plane handles the movement of a sensor node, i.e., detects and registers the location every time the sensor moves so that routing is always maintained. The third plane deals with scheduling tasks among sensor nodes in a given area of interest, i.e., sensors perform the sensing task at different times according to the task management unit. As a result, the three management planes allow sensors to collaborate in a power efficient way, maintain routing in a mobile network, and allow sharing resources among sensor nodes.

A Cross-layer approach is adopted in designing sensor network communication architecture, attempting to resolve the limitations in the layered approach discussed above. The new design proposes to have modules that involve, as part of their functionalities, communication interfaces to other modules in other layers, including non-adjacent ones Figure 2-4(b). For example, information available at the physical layer, such as signal strength of a neighboring node, can be of help to the routing protocol at the network layer in such a way to make different forwarding decisions accordingly. cross-layer design is anticipated to offer more flexibility and lead to optimization. Nonetheless, careful design is required; otherwise, it would degrade the system performance.

2.4 Challenges: Sensor Capability and Security

Different from ordinary computer networks, applying security in the field of wireless sensor networks faces special challenges or difficulties due to certain constraints.

This section summarizes sensor capabilities in terms of resources, communication facility, and nature of operation.

- ***Poor Resources*** (memory, processor, and power): as can be inferred from Table 1 above, sensors are deprived of the luxury of having strong resources, similar to all other networks, to run security algorithms, which demand a certain amount of resources - memory space for the code and data, processing power, and energy. Therefore, security algorithms code has to be kept small, which may involve modification and optimization to traditional security functions. For example, a sensor with 8 bit, 4MHz processor and 8KB memory, such as the Smart Dust, has the challenge of compiling together the operating system (TinyOS, 4KB), the core scheduler (178Bytes), and the application code. Therefore, adding on top of those components security algorithms can be challenging.

  Energy also restricts the freedom of implementing a large code or exchange long packets. Sensors feed on temporary power sources (i.e., battery), which cannot be replaced or recharged once deployed in a sensor network. Therefore, when adding a cryptographic function or protocol, the energy impact of the added code must be considered as to how that would affect the lifetime of a node. Consumption of extra Jules is directly related to the extra processing required to run the cryptographic function (e.g., encryption, decryption, signing data, and verifying signatures), to the transmission of overhead bits and data related to the security function, and to the energy required for storing security parameters (e.g., cryptographic keys).

- ***Unreliable Communication:*** communication between sensors is not reliable, mainly suffering from collisions, latency, and the connectionless nature of packets routing, which degrades heavily sensors security because the security of a network relies directly on a defined protocol, which, in turn, relies on communication.

Due to the broadcast nature of sensor networks, the chance of collision to occur is high. Increased collision result in an unreliable communication and hence a given security protocol may fail to run properly. The denser the network is, the more collisions are to be expected.

As the number of nodes increases, congestion becomes worse, routing paths become longer, and nodes will have to process more data. Those three factors incur more delay in the network. Consequently, synchronization among sensor nodes can heavily be achieved. Security mechanisms require exchange of information in a timely fashion strictly, e.g., distribution of cryptographic keys or prevention of reply attacks.

Error rate in wireless sensor networks by default is high, leading to packet loss and damage. Software developers are required to handle errors by incorporating the mechanism for that, such as error detection and correction. However, on one hand, such mechanisms entail more resources and incur overhead. On the other hand, lack of an error handling mechanism may result in losing critical security packets, e.g., cryptographic keys.

- ***Unattended Operation***: in most cases, once sensors are deployed, they are left unattended, behind enemy lines in some cases. Management, for example, may take place remotely. As a result, physically tampering with sensors is very likely to happen and detection is extremely difficult. It can be assumed that for a sensor being unattended for a long time, compromise is likely to have happened and security mechanisms have to take this into account.

2.5 Threats to a WSN

After deployment, a sensor network encounters many vulnerabilities and threats, to which employed protocols must be alert in order to mitigate those challenges. Threats extend from non-deliberate node breakdown, such as power failure or hardware or software crash, or adversarial attack, such as physical tempering or eavesdropping. The following list identifies those threats.

Passive Information Gathering: an intruder can place a device with a powerful receiver inside the network between nodes to eavesdrop on the ongoing traffic without interleaving.

Subversion of a node: that sensors are deployed in remote locations and are left unattended, node compromise is to be expected. An attacker is interested in uncovering the information stored in a sensor node, such as cryptographic keys, and possibly be able to inject false messages through this node, thereby becoming an insider attacker. For those reasons, nodes must be designed to be tamper resistant and react to any compromise attempts. For example, a node should delete its program memory content and its secret keys once node capture occurs.

False Node: in absence of security mechanisms, an intruder can add a false node to a sensor network. The intruder’s node is usually powerful and has the capability of impersonating a legitimate node. Being part of the network, the added node can disrupt the network operation by, for example, dropping packets, injecting false data, or modifying the contents of a message, hence corrupting messages integrity.

Node Malfunction: A node may become defective for ordinary reasons and, therefore, outputs wrong data. It is worse if a node that serves as an aggregation point or a cluster head becomes faulty. Sensor networks should involve the mechanism by which they can detect and eliminate damaged nodes.

Node outage: in addition to malfunctioning, a sensor may stop responding completely. If such a node serves as an aggregator, or cluster head, or simply as an intermediate node through which traffic is routed, then a sensor network should have the enough mechanism to detect such outage and, accordingly, provide an alternative node immediately.

2.6 Security Requirements

Sensor networks pose special security requirements due to the constraints discussed above (section 4). The choice of which security services to implement on a given sensor mainly depends on the type of application and its security requirements – availability, data confidentiality, data integrity, data freshness, and self-organization. Below is a discussion of each of them:

Availability:

Generally, availability can be defined as the survivability of a network. A network should consistently and continually provide the service that it promises despite the existence of any attacks. The reason of adding cryptographic functions is to protect the sensor network from attackers whose goal is to bring down the network or at least decrease its lifetime. However, involving cryptography is not free. In fact, modifying cryptographic tools to fit sensor networks and achieve the same level of security as in other networks unfortunately adds to the cost. For example, some of the techniques propose for modification additional communication, reuse and adjustment to the current security functions code, and/or centralized schemes instead. Such techniques impact the availability of sensors and sensor networks because additional computations and communication consume additional energy, thus decreasing sensor lifetime. Centralized schemes are also a great threat to the availability as they introduce a single point of failure.

Data Confidentiality (Privacy):

Confidentiality means restricting access to information to only those authorized to have access. In certain sensor network applications, confidentiality becomes big concern. For example, it could be of high importance (e.g., in military applications) to secure sensor readings from leaking to other neighboring networks (e.g., adversary's network). In another example, nodes may have to exchange sensitive data, such as cryptographic keys, for which secure channels are required. Furthermore, to prevent traffic analysis attacks, information such sensors identities and public keys have to be encrypted.

The standard technique for achieving confidentiality is through encryption/ decryption mechanism using secret keys that only legitimate parties posses.

Data Integrity

The term data integrity refers to the mechanism that guarantees that data has not been altered in transit. Without this security requirement, data is not safe as any adversary can change the data or fake some messages. In certain applications, confidentiality is not as important as data integrity, allowing an adversary to eavesdrop and learn about the real data but not to change it.

The standard method for achieving data integrity is using a hash function, which produces a digest value that uniquely represents a packet. A change in even a single bit would indicate data forgery.

Source Authentication (Data Authentication)

By definition, data authentication allows a receiver to verify that the data is truly sent by the claimed sender. Obviously, an adversary can replace the whole packet stream by injecting additional packets with rehashed digests. In many communication sessions, such as administrative tasks (e.g., control commands), the receiver needs to confirm the data origin before applying received information in any decision-making process.

Data authentication is achieved by using keyed hash functions, also called Message Authentication Code (MAC). Two nodes possessing a shared symmetric key can achieve data authentication easily by computing (at the sender side) and verifying (at the receiver side) MAC values for all communicated data. In the case that authenticating a broadcast message is required, a different mechanism for example, asymmetric cryptosystem or µTESLA protocol [14], must be applied.

Data Freshness

Data freshness brings up the issue of how recent the data is. Assuming data confidentiality and integrity are achieved, Networks cannot afford not ensuring freshness of each message so that old messages are not replayed. This requirement is especially important when there are shared-key strategies employed in the design.

Different packets are typically processed with different keys, but distributing new keys every time is hard to realize. For this reason, launching replay attacks seems to be appealing to attackers and can cause confusion in the network especially if a node is unaware of key change.

Typically, to achieve data freshness and prevent replay attack a new nonce (a randomly generated number), or another time-related counter, is attached to the message.

2.7 Operational Paradigms and Corresponding vulnerabilities

It is important at first to make a distinction between sensor networks and their ancestors, ad hoc networks. The two are usually confused for the similarities they share. Both, for example, take advantage of multi-hop networking for communication among nodes lying outside one another’s communication range. However, a number of dissimilarities can be observed between the two. Sensors are more resource limited, are usually deployed densely, are more prone to failures, and are several orders higher in number. Furthermore, sensor nodes often observe the same or correlated environmental events, which necessitate in-network processing to eliminate duplication. In terms of routing, in ad-hoc networks, communication can take place between any pair of nodes; whereas sensor networks support only certain communication patterns, such as the following:

1. Many-to-one: clustered sensors send their readings to an aggregation node or to a base station.
2. One-to-many: usually this takes place in the form of broadcasting a query, submitted by a base station for example.

3. Local communication: nodes within the same radio range may exchange local information, for example, in the process of forming clusters.

Sensor networks vary in their architecture depending on the application needs, i.e., certain components may be installed in different ways. However, typically, a sensor network, similar to the one shown in Figure 2-5, consists of certain components, namely, sensor nodes, a sensor field, a sink, and a task manager, all defined below:

- Sensors or motes compose the heart of the network. They function as sensing elements to read environmental data and as routers to deliver information to the sink node. The number and density of sensors deployed in a location are determined by the application in question. A few sensors, or sometimes hundreds to several thousands of sensors, are required.
- Sensor field defines the area of interest, i.e., the area in which a physical phenomenon is to be monitored. Sensors are deployed in such a way to cover that area.
- A sink, called an aggregator sometimes, is a special sensor usually with enhanced capabilities and with a specific task. There can be multiple sinks in a sensor network. They receive messages from multiple sensors, process this received information, and combine messages in order to reduce the total number of messages in the network, hence optimizing the network overall energy consumption.
- Task manager, also called base station and home server, is a centralized control station from which queries are initiated. Usually it is remote from the

area monitored. Upon a query sent from the task manager and broadcast to sensors in a certain area, the sensing process takes place and results are streamed back through sensors and sinks to reach the task manager. Some sort of fast communication medium such as Satellite and/or the Internet may be used.

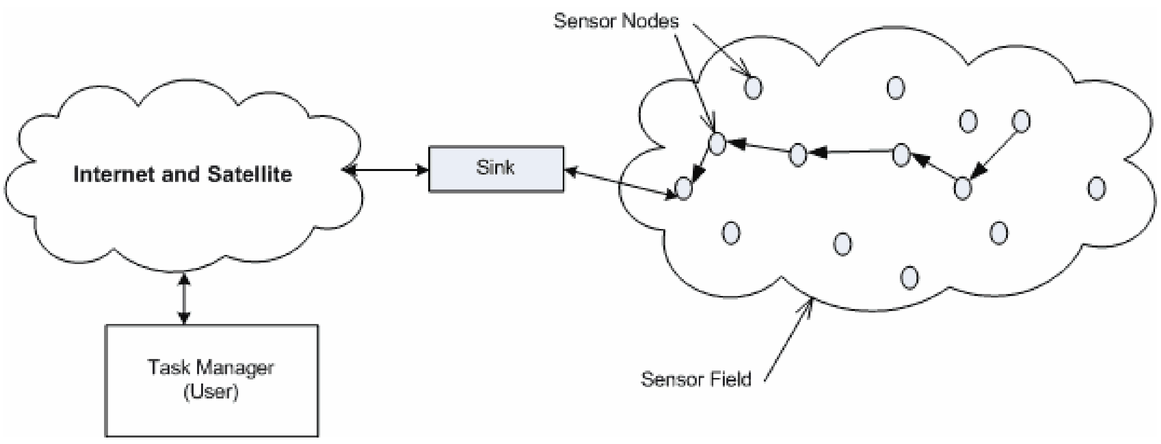


Figure 2-5: Typical Sensor Network Structure

The discussion above suggest that sensor networks follow different models of operation (or paradigms) and that security requirements differ according to the type of network. Some operational paradigms can be as simple as sensors performing measurements and pass them directly to a central station. Others involve complex mechanisms, such as routing and/or aggregation. Because of the variety of operation paradigms, designing a unified security model is not feasible. Security goals will depend on the type of network, security requirements, and the assumptions about the adversary's target and capabilities. For example, is the network an aggregation one? Should the messages contents be encrypted or is messages integrity enough? What is the required level of robustness? How powerful is the adversary in capturing and compromising nodes? Many more questions can be asked to help in maintaining proper security measures.

The following is a categorization and a brief discussion of the operational paradigms and their corresponding vulnerabilities [15].

Collect and Transmit: this paradigm represents the simplest sensor network model, in which sensors are only concerned with their transmittal to a within-range base station. Thus, multi-hop routing or co-operation among nodes is not to be concerned about. Sensors can perform their tasks either in a periodic manner or event driven.

This paradigm is vulnerable to a number of attacks. In denial-of-service, for example, an attacker tries to jam the radio signal and cause collision. Eavesdropping on traffic and spoofing packets are also possible. Additionally, similar to all other paradigms, physical attacks, meaning capturing and compromising nodes, are very possible.

Forward: different from the previous paradigm, nodes co-operate among one another as every sensor transmits its measurement to one or more neighboring sensors, which, in turn, propagate the same data to additional sensors. Retransmission continues at every node until the data is received at the central station eventually.

All the vulnerabilities mentioned in the Simple Collection and Transmittal paradigm and others apply to the Forward paradigm. Because this paradigm is more sophisticated with the new functionalities, such as routing, vulnerabilities are more. For example, nodes on the way of traffic may corrupt data and lie about measurements. Moreover, malicious nodes may flood the network intensively with messages, consequently causing sensors to exhaust their resources (e.g., energy) in trying to forward all received data.

Receive and Process Command: in this paradigm, forwarding of packets also takes place towards the base station (many-to-one communication). Additionally, propagation of a special type of messages, called commands, move in the opposite direction, from a controller or base station to the network nodes, either in a broadcast form to multiple nodes (or probably the entire network) or unicast form to a specific node (one-to-many communication). Those commands are required to configure or reconfigure the network and hence control the amount of data in the network, e.g., specify or modify the periodicity of performing a certain measurement.

One vulnerability to mention on top of all the discussed vulnerabilities in the previous paradigms is the threat of impersonating the controller and thereby disseminating misleading commands.

Self-Organization: because of the scalability to hundreds or thousands of nodes requirement, sensors need to self-organize themselves as to determine the topology every time it changes, adding or removing nodes. Clustering is one technique to achieve self-organization, in which certain nodes that are believed to be more powerful according to some criteria are elected as cluster heads. The elected nodes will function as routers as they receive packets from the same cluster sensors (intra-cluster communication) and pass those packets to other cluster heads (inter-cluster communication) with the intention that the packets will arrive at the base station.

The paradigm here depends heavily on the ability to route traffic along the routing nodes. As such, on top of all the vulnerabilities mentioned so far, attacking the routing protocol can harm the network greatly; for example, create routing loops, leaving the network stranded.

Data Aggregation: The large number of nodes may lead to enormous amount of data in the network, causing the network to degrade performance and shorten its lifetime. For those reasons, there exists nodes whose responsibility, besides sensing, is to aggregate or combine messages coming from different nodes, resulting in much less traffic.

In addition to the common vulnerabilities (impersonation, lying about data, dropping packets deliberately), authentication specifically is more difficult to achieve in this paradigm because it is the aggregator now, not the base station, who must ensure senders authenticity before processing the received data. In Chapter 3, data aggregation and corresponding security protocols are presented and explained in detail.

Flexibility and Adaptation: this paradigm adds to the intelligence of a sensor network as sensors decide on their own what to do next based on their measurements. For example, a node may decide not to do inter-cluster communication if its power becomes less. Similar to the aggregation paradigm, that nodes make decisions and manipulate data require trusting nodes and authenticating them, which is very difficult in reality.

CHAPTER 3 LITERATURE REVIEW

SECURITY AND DATA AGGREGATION IN WIRELESS SENSOR NETWORKS

This chapter addresses two important issues in sensor networks: security and data aggregation. A review of literature of research work conducted to address these two issues is reported. In summary, this chapter discusses the following topics:

1. Introduction to the Byzantine behaviour problem as originally introduced by Lamport et al [16]. Introduction to cryptographic systems, symmetric and asymmetric, their relevance to SN; a discussion on their incurred energy cost implication is also provided.
2. Introduction to wireless sensor networks vulnerabilities and attacks, and a summary of various existing secure protocols.
3. Overview of the concept of information aggregation and the different techniques that are used to aggregate data in SN.

3.1 Byzantine Behaviour

Definition: in networking [16,17,18], the term Byzantine refers to certain nodes in the network that exhibit abnormal behaviour. Specifically, it addresses two kinds of such behaviour: malfunctioning and malice. In the first type, nodes might operate incorrectly due to power outage or wrong configuration. In the second type, malicious nodes might provide incorrect information, lie about routing information, discriminate between different kinds of traffic by forwarding some and dropping others. It is assumed that malicious nodes can disrupt the network in various ways while staying undetected.

The concept of Byzantine behaviour was first introduced by Lamport et al. [16], and in which they called the “Byzantine Generals Problem.” Below is a summary of the problem, a description of the way it was originally addressed, and the follow up citations.

Lamport et al. [16] pioneered the work to addressing the problem of consensus in distributed systems, presenting it in terms of a story known as the “Byzantine Generals Problem.” Briefly, in that problem, there are a number of processors, n, known as generals. They form a mesh network such that any pair of the generals can communicate privately through channels called “oral messages.” One of the generals is assumed to have a special role (commander), in that he sends a binary value (command) to the n-1 generals, called lieutenants. The binary value corresponds to “attack” or “retreat.”

The idea is that some of the generals, including the commander, might be traitorous. A traitorous general may exhibit a devious manner, such as by failing to send messages, sending malformed messages, sending forged messages, or sending contradictory messages to different nodes. However, the desired behaviour is that the honest generals (non-faulty processors) are supposed to reach an agreement with the following three conditions:

- Termination: Eventually each honest general sets its decision.
- Agreement: the decision value is the same for all the honest generals.

- Integrity: if the commander is honest, then all honest generals decide on the value that the commander initiated.

The concept was appreciated by the computer science community and inspired its members to generate hundreds of papers and theses. Some of the results are summarized here:

- No solution can exist if less than or equal to 2/3 generals are honest. [19].
- If t faults are to be tolerated, at least t rounds are required for any deterministic algorithm [20].
- In an asynchronous system, the problem is unsolvable [21].

As a result of these findings, bringing the notion of consensus to networks seems more challenging because networks are not necessarily fully connected and communication in reality is asynchronous, not to mention that the information to be delivered is not as simple as just attack or retreat. However, cryptography is promising to make the problem appear tractable. The next sections show how such a network can be designed and implemented with the existence of Byzantine nodes.

3.1.1 Byzantine Behaviour and Security

Byzantine behaviour is usually used in the context of security. However, the distinction between the two should be made clear. Byzantine failure models were not originally meant to deal with malicious attacks. Assumptions such as $n>3t$, for example, are hard to justify when assuming malicious intent, as nodes in this case may collude, resulting in nondependent node failures. Byzantine behaviour does not address the issue of data secrecy either, i.e., data confidentiality, integrity, and authentication. As a matter of fact, Byzantine algorithms require replication of messages, which means that confidential data can at increased risk. The key and the only solution for data secrecy is to involve cryptographic models.

Byzantine models and security models should be thought of as complementary components for a robust system, that is, one that ensures both network survivability and data secrecy.

3.1.2 Levels of Robustness

All networks are prone to failures as a result of corrupted nodes or disconnected communication links. With Byzantine nodes, failures happen because of abnormal functioning, whether deliberate or not. Algorithm robustness can be categorized in the following four levels:

Simple Failures: Some networks ignore the possibility of node failure. As a result such networks continue to operate in the existence of malfunctioning nodes but incorrectly. This situation makes maintenance tasks complicated because manual intervention is required to bring the network up again.

Self Stabilization: Algorithms under this category show a slight improvement over the previous category. They guarantee correctness as long as the network is free of malfunctioning nodes, which can be achieved only after such nodes are removed.

Detection of Byzantine Behaviour: Algorithms designed with this property are not able to work properly with the existence of Byzantine nodes; however, they make identifying a corrupted node easy. Combining this property with self-stabilization enhances the robustness of such algorithms.

Byzantine Robustness This property demands the highest standard of network survivability: a network continues to work correctly even with the existence of faulty nodes. However, Byzantine detection is not necessarily achieved.

Recently, a number of papers, such as [23,24,17,18], have shown the need for integrating an algorithm, such as routing, with Byzantine robustness to mitigate internal attacks and therefore ensure network operability at all times.

Yu et al. in [23], secure their routing algorithm against external and internal (Byzantine) attacks. Specifically, asymmetric cryptographic functions are used to defend against external attacks. These functions assume that each node holds a private/public key pair. Nodes learn their neighbours through signed route discovery messages. Then each node creates a set of shared keys that it can use with different nodes at different levels of neighbourhood - 1-hop, 2-hop, or more. They defend against Byzantine attack by using message and route redundancy. After the route discovery phase, a route can be selected among disjointed routes. A compromised node, at worse, can drop packets, but because of the redundancy of the packets, the destination expects to receive the same message from different nodes. Not receiving it from certain nodes suggests Byzantine behaviour. A message is broadcast upstream and downstream to let both the source and destination learn about any malicious nodes. Each node builds a local trust repository for the nodes it knows, based on the observed behaviour. Accordingly, a path containing a node with a low trust value is excluded.

Awerbuch et al., in [24], claim a Byzantine resilient routing protocol that consists of three phases. The first phase is route discovery with fault avoidance, which is achieved through flooding and using cryptographic primitives, finding thereby a number of paths with different weights between the source and destination. The second phase is Byzantine fault detection, which is realized by probing the nodes composing a path. Nodes on the path, including the destination, return acknowledgement messages to the source. Failing to do so lets the source mark the path as faulty. Through the probing mechanism, the source can also detect the faulty nodes and is able, in phase three, to assign weights to the discovered links.

This protocol, in fact, cannot scale to a large network or a hierarchal one. The number of acknowledgement packets to a single request is very high, even in the absence of an attacker.

Perlman's work objective in [17] is to secure the network layer, preventing malicious nodes from generating incorrect routing information. The security model used is centralized and based on public key cryptography, that is, by assuming that one node, T, is trustworthy. T holds all the public keys of the other nodes. If a node requires a key for communication with another node, it sends a request to T.

Because of the high number of messages involved, asymmetric cryptography, and the need for a cache for every link, this protocol requires powerful machines for implementation.

3.2 Securing Wireless Sensor Networks

3.2.1 Cryptography

The severe resource constraints limit the freedom in applying cryptographic primitives in sensor networks. For example, some researchers argue that asymmetric cryptography (or public key) is not feasible at all because of its high memory and power consumption requirements [14, 26]. A hardware solution, such as the one in

[25], to public key cryptography in sensor networks seems to be costly and not practical in a large network. Symmetric cryptography is the alternative. As has been argued by Wagner et al, in [27], symmetric system tools provide an acceptable level of security, making them preferred over those for asymmetric systems. Below is a brief introduction to cryptographic systems and illustration of some work done in this direction.

Symmetric Cryptography

Symmetric cryptography is base on two communicating parties sharing the same secret key (Figure 3-1). The sender uses the secret key to encrypt a message (or any cryptographic function), and the receiver uses the same key to decrypt the message. Symmetric cryptography satisfies the sensor network resource constraints, and it is fast and does not consume much power. The problem with symmetric cryptography, however, is the difficulty of secret key distribution and management, for example, how to supply two sensors with the same key without revealing it to anybody. Figure 3-1 shows a visual representation of symmetric cryptography.

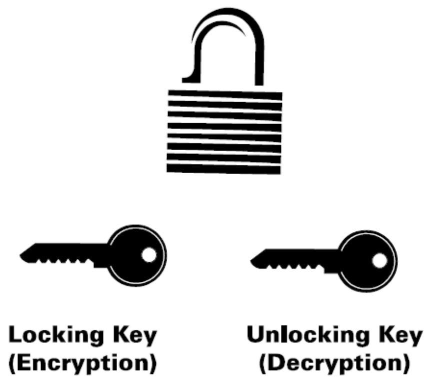


Figure 3-1 Symmetric Key Cryptography

Asymmetric Cryptography

Asymmetric cryptography was introduced initially to solve the key distribution problem. Asymmetric cryptography is different in that every communication party possesses a pair of keys; one is kept secret and called the private key and the other is distributed and called the public key (Figure 3-2). The sender uses the receiver's public key to encrypt a message, and the receiver side uses its own private key to decrypt it. Authentication is also possible through this system. The sender can send a tag by hashing a message by its private key, and the receiver can verify the message using the sender's public key. Unfortunately, asymmetric cryptography requires very high computational power because it is based on complex mathematical models (e.g., factoring large prime numbers).

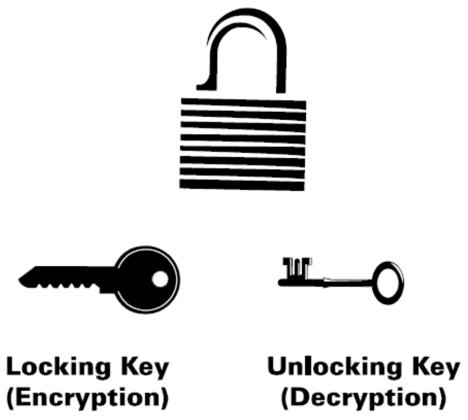


Figure 3-2 Asymmetric Key Cryptography

A good explanation of basic cryptographic primitives, based on symmetric and asymmetric systems, can be found in [28] by Menzes et al. Such primitives include Hash Function, Pseudo-Random Function (PRF), Message Authentication Code (MAC), encryption, decryption, and many other primitives. Implementing a cryptographic function, though, is not straightforward. Decisions of whether to use symmetric or asymmetric solutions and what specific cryptographic algorithm to use must be made first. Wei et al. [29] present a comparative analysis of various symmetric ciphers, such as RC5, RC6, Rijndael, MISTY1, KASUMI, and Camellia. Gura et al. [30] provide an experimental analysis of RSA and elliptic curve cryptography (ECC) on sensor networks.

3.2.2 Energy Constraint

Being a micro-electronic device, a sensor node can only be equipped with a limited energy source, e.g., less than 0.5Ah and about 1.2V battery. Energy, therefore, is the scarcest of all sensor resources. Each milliamp consumed brings the sensor one milliamp closer to death. Thus, to expand the lifetime, nearly every aspect of sensor networks must be designed with power in mind. For example, when working at full power, a typical sensor, such as MICA2, can operate only for two weeks or so before it dies. The goal is to elongate its lifetime to months or years.

For the sake of analysis, power consumption is directly linked to the tasks a sensor node performs. It detects events, performs internal processing, and finally transmits data. Thus, power dissipation can be divided into three categories: sensing, data processing, and communication. Of the three categories, a sensor node expends maximum energy in data communication, transmission and reception. For example, the transmission of each bit is equivalent in power consumption to executing 800 – 1000 instructions. Consequently, as far as security is concerned, a system that uses cryptography incurs extra overhead in the length of messages transmitted and also extra use of the memory and processor. In sensor networks, as opposed to conventional networks, a packet is very small, typically 30 bytes. A cryptographic primitive that adds 8 bytes, for example, incurs an overhead of almost 25%.

Several studies have been conducted addressing energy consumption problems and suggesting solutions for the use of security services in sensor networks. Perrig et al., in [322], develop μTESLA and SNEP security tools (discussed in section 3.2.4). In trying to keep the overhead small, they suggest the use of a stream cipher for encryption because that results in the same encrypted message size as that of the plaintext. However, MAC adds an overhead of around 8 bytes to a 30-byte message. As a result, for an encrypted and authenticated message, the cost reaches around 30% overhead. This overhead is mainly due to the transmission of extra data rather than computational cost, which is around 3% only.

In [32], Wagner et al. also develop two security mechanisms, namely, TineSec-Auth (for authentication), and TinySec-AE (for authentication and encryption). For the encryption, they have chosen to use Skipjack, which is a block cipher but thought to be more secure than stream ciphers. For authentication, they employ CBC-MAC, whose code is partially used in skipjack. As far as energy is concerned, TinySec-AE includes 5 bytes of overhead, causing 13.6% of energy consumption increase. On the other hand, TinySec-Auth increases the packet length by 1 byte, leading to 2.6% of energy consumption overhead.

More work is presented by Kelner et al. in [22], in which they study several security algorithms in terms of the energy consumption caused by radio, CPU, and memory. They implement them on MICA2 sensors and provide actual energy consumption measurements according to the formula $E=V\times I\times\triangle T$ (where E is energy in Joules, V is voltage in Volts, I is current in Amperes, and $\triangle T$ is time duration in Seconds). Their tests include SkipJack, RC5, RC6, TEA, and DES. They conclude that DES is unsuitable to sensor networks because of the high energy consumption consequence, while SkipJack, RC5, and TEA are possible candidates as far as energy is concerned.

3.2.3 Types of Attacks

Sensor networks are highly susceptible to a variety of attacks. In fact, many of the attacks that target wired networks and ad hoc networks are considered possible threats to sensor networks as well. Additionally, some of the attacks are specific to wireless sensor networks to disrupt, in particular, their functionalities, e.g., routing, time synchronization, sensing. Attacks can be classified based on the power of the attacker's devices and on whether the attacker has gained access to the network [2]. In the first case, either mote-class or laptop-class equipment can be used. Mote-class attacks assume that the attacker uses malicious sensor nodes similar in their capabilities to the ones in the targeted network, causing harm to only a few nodes at worst, for example, by jamming the radio signal within the malicious node's range. Laptop-class attacks, on the other hand, are more harmful, as they may disrupt the entire network using only one powerful machine, such as a laptop or an equivalent machine with a strong transmitter and sensitive antenna. Attackers can be seen also as outsiders or insiders. In the former, the attacker does not have access to the network. However, when this attacker manages to compromise a legitimate node or to steal the security keying material, the attacker becomes an insider, leading to more dangerous attacks, especially when laptop-class devices are used.

The following is a collection of well-defined threats drawn from different papers.

Denial of Service (DoS): Many attacks can be described in terms of DoS. However, this term is usually linked with attacks that mainly waste resources or disrupt services with little effort by an attacker. The most common such attacks target in specific the physical layer and data link layer. At the physical layer, sensor networks may experience jamming [34], which can be launched simply by the transmission of radio signals that interfere with the sensor network frequencies. As a result, communication among nodes fails as long as the attack is maintained. Physically tampering with nodes is also a serious threat. For example, the attacker can damage the sensor node, replace code, or spoof the sensing element measurements to provide erroneous readings. At the data link layer, causing collisions is the main threat, and is described also as link layer jamming. An adversary tries to corrupt a transmitted data packet or acknowledgement packet by sending packets at the same time. Some data link protocols, consequently, require retransmission of failed data packets, leading to battery depletion. Corruption of ACK packets also lead to costly exponential back-off time in certain MAC protocols.

As mentioned above, other attacks targeting network, transport, and application layers could be classified under DoS. Nevertheless, a list of different attacks is introduced separately below.

Spoofed, Altered, or Replayed Routing Information: [2] In attacks of network layer, network operability can be disrupted very easily if the routing information exchanged between the nodes is targeted. For example, this attack can result in routing loops, rejecting or attracting traffic, partitioning the network, or increasing delay between source and destination.

Hello Flood: [2] This attack targets certain routing protocols that rely on broadcasting HELLO packets to learn about the neighbouring nodes, e.g., TinyOS beaconing protocol, and clustering based protocols, such as LEACH. A malicious node with high transmission power can send, record, or replay HELLO messages to reach sensor nodes in the area. The goal is to create an illusion of being a neighbour to many nodes in the network. Consequently, the nodes mark the adversary as a neighbour even though they cannot communicate back to the adversary, which leaves those nodes stranded. Authentication does not solve the problem completely because the adversary may also replay legitimate packets. Rather, bidirectionality of the communication should be checked.

Selective Forwarding/ Black Hole: [2] Routing algorithms often assume that nodes are trustworthy. A malicious node, for example, working as part of the network, can selectively drop certain types of traffic and forward others. Black Hole is the extreme case of selective forwarding, in which a malicious node blocks the traffic flow by dropping all the packets. However, routing techniques are usually able to discover traffic blockage in a given path and reroute packets accordingly. Some examples of the protocols affected by these attacks are directed diffusion, TinyOS beaconing, and geographic routing protocols, such as GPSR [35].

Acknowledgement spoofing: [2] A smarter way of mounting Selective Forwarding or Black Hole attacks can be achieved through acknowledgement spoofing. In the routing protocols that choose the next hop based on reliability issues, the adversary can convince the sender that a weak or even a dead link is a strong one by faking link layer acknowledgment packets, which leads the sender to assume that the link can be trusted and used for future sessions even though the real data packets are actually lost.

Wormhole: [36] This attack requires at least two colluding malicious nodes, one of which is located in the area of interest (usually close to the sink), tunnelling messages to the second one over a fast link. A Wormhole attack's goal is to disrupt routing algorithms by convincing two multi-hop distant nodes that they are only one hop apart. Furthermore, it makes selective forwarding and eavesdropping more achievable because all traffic is attracted to pass through the malicious nodes. Rumor routing and minimum cost forwarding are some of the routing protocols that can be attacked this way.

Sinkhole Attacks [2]: in this attack, the adversary mounts a device (usually a powerful one, such as a laptop) that appeals to the neighbouring nodes with respect to the routing algorithm so that all traffic is pulled towards the malicious device (as the attack name suggests). Since the traffic flows through the attacker's node, tampering with packets becomes much easier, and various sorts of attacks become possible (e.g., black hole, selective forwarding, modifying data).

A Sinkhole attack is particularly easy to create because of the routing nature in sensor networks whereby traffic from everywhere is directed to the same common destination. Routing algorithms always look for the best path. The sinkhole node can simply claim to provide a single high quality route.

The Sybil Attack [37]: To defeat the existence of Byzantine nodes, many routing algorithms maintain fault-tolerant schemes, such as finding multiple disjointed paths. The Sybil attack is known to defeat such mechanisms because in such an attack a single malicious node presents itself with multiple identities to other nodes in the network, leading to the illusion that multiple paths are present.

Furthermore, geographic routing protocols are also susceptible to Sybil attacks. For example, location-aware routing algorithms require nodes to exchange information about one another's position for use it in the packet's address. Sybil attacks can simply announce multiple locations simultaneously, causing the routing algorithm to fail.

Time Synchronization Attacks: Song et al., in [58], present delay attack, which aims at corrupting time synchronization protocols in the transport layer so as to create a gap with respect to time between nodes. In many sensor network applications, such as object tracking, message ordering, data aggregation, time synchronization is an essential part of the application success. For example, in object tracking, unless sensing nodes are able to record the time and location of a moving object precisely, there will be no accurate information. Therefore, attacking the time synchronization scheme appeals to an adversary as doing so destroys sensor network applications. Sensor nodes achieve time synchronization by exchanging special messages. Thus, in defence, synchronization messages are protected by means of cryptography. However, insider attackers are still a source of threat.

Stealthy Attack: This strategy was first addressed by Perrig et al. [64]. In a sensor network in which aggregation of sensor data is required, sensors, including the aggregators, may lie about their measurements or their aggregation results. This attack works at the application level to prompt the base station to accept wrong information.

Traffic Analysis: This term usually refers to the process of inferring information about the communication of an encrypted network. Sensor networks follow a special pattern of communication, which enables attackers to learn, for example, where information is destined. Unencrypted routing information and addresses make traffic analysis much easier.

Deng et al., in [65], address the problem of detecting a base station through traffic analysis; once a base station is detected and so attacked, an adversary can render the whole network useless. One way to reveal the base station's location is through a rate monitoring attack. The underlying idea is that the number of packets increases the closer they get to the base station. All the attacker needs to do is trigger an event and trace where the packets propagate.

Table 2 lists the types of attacks categorized by the layer they target.

Table 2: Types of Attacks

Layer	**Attack**
Application Layer	Insider attacks
Transport Layer	time desynchronization, flooding
Network Layer	Selective Forwarding, Black Hole, Ack. Spoofing, wormhole, sink hole.
Data Link Layer	Collision, Exhaustion
PhysicalLayer	Jamming, Tampering

3.2.4 Defending Mechanisms and Secure Protocols

From the discussions above, it is clear that although employing security measures is challenging in such a resource-constraint environment, doing so is essential for the operability and survivability of sensor networks. Sensors must have the ability to authenticate a sender's identity (source authentication), receive all messages intended for it (availability), ensure that messages are not altered or repeated (integrity and freshness), and, optionally, prevent an adversary from revealing the contents of messages (confidentiality). Finally, access to the network must be restricted only to legitimate nodes.

Research has been growing tremendously in this area recently, trying to develop secure protocols for wireless sensor networks. Many of the researchers propose adding optimised security mechanisms, e.g., cryptographic tools, on top of the existing protocols. They argue that such techniques produce sufficiently secure algorithms with less cost as they reuse existing sensor network protocols. On the other hand, others stress that unless protocols are redesigned from the ground up, no security can be fully guaranteed. Obviously, some of the old protocols that were not designed with security in mind are hard or sometimes impossible to integrate with security measures. Another difficulty is linked to attacks coming from the inside. Insider attackers require smart mechanisms that are beyond the capability of cryptography. Finally, different attacks target different layers as can be seen from the discussion above. A secure protocol designer has to consider the layer in question (Table 2).

All in all, secure protocols must achieve security requirements, taking into account all the hardware obstacles, the existence of insider attacks, and the proper layer to be secured against a certain attack. In the following sections, defence mechanisms ([33],[2]) and several secure protocols suggested in the literature are presented.

Avoiding jamming attacks requires design tuning and is usually costly. Ideally, the base station should learn about jamming in a certain area, e.g., by letting the jammed nodes inform their non-jammed neighbours and those, in turn, pass this information on to the base station. Many techniques are suggested, for example, spread spectrum, switching to a low power cycle, changing to Infra Red or optical instead of radio frequency. However, all these solutions require extra cost.

Tampering can be avoided using tamper resistant packaging at extra cost. Otherwise, sensor nodes should be programmed in such a way as to erase sensitive data if a node is captured.

Collision as an attack is almost impossible to mitigate. The best way is to design the data link layer to be able to detect the collision attack and employ stronger error correction schemes.

Most outsider attacks can be prevented through simple cryptographic means, e.g., link layer encryption and authentication. Selective Forwarding, Sinkhole, Black Hole, Sybil attacks, acknowledgment spoofing, and de-synchronization are not possible because the attacker cannot join the network. Nevertheless, if the attacker manages to compromise some nodes, another level of protection will be necessary. For example, multiple paths should be provided to reduce the impact of selective forwarding. With Sybil attacks, nodes have to ensure the identity of other nodes they are in contact with, for example, by having unique keys so that a compromised node cannot claim different identities.

In the case of Hello Flood attacks, cryptography does not solve the problem because an attacker can replay an encrypted Hello packet in another area of the network and still create the same harm. To resolve the problem, nodes should verify the bi-directionality of a link before marking the station that is sending Hello messages as parent.

Wormhole and Sinkhole attacks seem to be the most difficult ones to mitigate because they both use out-of-band communication channels. Karlof and Wagner, in [2], suggest that geographic routing shows promise in defending specifically against these two attacks because packets are routed towards the physical location of the base station. If an out-of-radio-range route is advertised, nodes will mark that route as an attack.

Perrig et al., in [31], pioneered the construction of symmetric cryptographic security tools optimised for sensor networks, Secure Network Encryption Protocol (SNEP) and the “micro” version of Time, Efficient, Streaming, Loss-tolerant Authentication Protocol (μTESLA). These two tools can be seen as the building blocks for sensor protocols and applications.

SNEP provides data confidentiality, two-party authentication, and weak and strong data freshness. Confidentiality is achieved through encryption of data concatenated with a counter value to ensure semantic security (different cipher texts for the same plain text). Two-party authentication and data authentication are realized by a MAC function that takes as input the shared key between the two nodes and a counter value. This counter value also has an important role in preventing replay attack.

μTESLA provides authenticated broadcast messages. In sensor networks, the base station is required to send packets to all nodes, e.g., routing beacons, queries, or reprogramming of the entire network. Using μTESLA to send an authenticated packet, the base station hashes the packet with a key that is not known to any node, including the recipients and, of course, the adversary. Nodes buffer the packet for a certain time until the key is disclosed by the base station. Through a key chain mechanism, nodes can verify the correctness of the disclosed key.

One problem with their proposed protocols is the use of a master key that is distributed to the entire network at the initial setup. Such a scheme weakens the security level because compromise of the master key may lead to compromise of the entire network.

Similar to SPINS work, TinySec [32], proposed by Wagner et al., secures the link layer in sensor networks; that is, it prevents unauthorized packets from joining the network even for a single hop. TinySec, specifically, works on a TinyOS operating system and supports two security options: authenticate encryption (TinySec-AE), in which data is encrypted and then the whole packet is hashed with MAC, and authentication only (Tiny-Ath), in which only MAC is computed for the packet. The authors' goal is to achieve the minimum overhead possible in terms of the secured packet size, which hence leads to less power dissipation. The default size of a packet in TInyOS is 36 bytes. They managed to create an authenticated message of only 1 byte overhead and an encrypted and authenticated message of 5 bytes overhead.

Karlof and Wagner, in [2], conduct an in-depth study of the current state of security in sensor networks. They investigate all possible attacks against routing protocols and suggest defense mechanisms for them. Furthermore, the authors analyze several common routing algorithms, such as TinyOS, Directed Diffusion, Geographic routing, LEACH, concluding that all these protocols are insecure and pose significant challenges. They recommend, therefore, designing routing protocols with security in mind from the start. Cryptography (link layer encryption and authentication) helps achieve a good level of security in routing algorithms; however, it is not the complete solution as can be proven in the case of Sinkhole attacks, Wormholes, and insider attacks.

Srivastave et al., in [66], define three levels of communication security, which map to three classes of data: mobile code, node location, and application specific data, ordered from high security sensitivity to low. Each level of security corresponds to a different level of security strength. The goal of layering security is to save energy on cryptographic algorithms since they consume different amounts of resources.

Avancha et al., in [38], move up the protocol architecture to address security in the application and network layers. Specifically, they develop techniques to secure certain routing and data movement approaches in directed diffusion [39] and SPIN [40]. The specific application considered is perimeter protection, which involves a lot of monitoring and data transmission. Added to symmetric cryptographic tools, their technique detects aberrant nodes for elimination.

An issue yet to be addressed is the possibility of achieving end-to-end security. Data originating at a certain node should be received at the base station intact. If such data is encrypted, for example, then it should be decrypted nowhere but at the base station. Nevertheless, in-network processing is important for efficient data handling as to reduce redundant packets. In this context, the next section addresses information aggregation and security issues.

3.3 Information Aggregation

Sensor networks typically consist of a large number of nodes. Hundreds or thousands are common. As the size of a network grows, it is obviously expected that the amount of data will grow proportionally. Sensors, due to their computational constraints, are not able to route or process such an enormous amount of traffic, but luckily do not need to do so. Smart techniques simplify the task by providing a summary of the results that a number of individual sensors come up with. For example, sensors that watch the same physical phenomenon are likely to provide similar information. They can send their raw data to a single node, where summarization can take place.

In other words, a group of sensors send raw information to a specific node that takes care of aggregation and provides smaller-in-size and more meaningful messages. This node is called an aggregator. The next step is to deliver this message to a central station as the final stop.

Aggregation is quite useful in reality. It offers a better utilization of bandwidth and battery by reducing the number of messages travelling around in a sensor network; it leads to more accurate, reliable, and meaningful data, and it helps prolong the network lifetime. However, aggregation is particularly vulnerable to attacks. At the level of the regular nodes, the existence of Byzantine nodes can lead to inaccurate aggregation results. What is worse is that when the aggregator itself is compromised, then the whole sensing task becomes meaningless. Therefore, secure information aggregation techniques are crucial to the success of a sensor network.

In the context of aggregation, the next section is an introduction to various ordinary techniques proposed for achieving data aggregation in sensor networks, followed by another section addressing security issues and enumerating secure aggregation techniques.

3.3.1 Data Aggregation Techniques

Before discussing security in this field, it is important to consider data aggregation techniques in general. The first underlying concept is clustering, which is proposed in the literature to allow expansion of sensor networks to a large number of sensors and to preserve energy to the maximum. In clustering, some nodes are elected as cluster heads, whose responsibility becomes summarizing or combining messages coming from regular sensors in the corresponding clusters, thereby providing less detailed information to distant nodes. [41[42][43][44] are examples of such clustering algorithms, and are summarized below.

The Voting-based Clustering Algorithm (VCA) [41] selects cluster heads by considering two physical properties related to the nodes themselves, the residual energy, and to the network, the topology. Nodes are initially assumed to learn about the existence of neighbours and their corresponding residual energy through frequently sent beacon messages. The algorithm lets each node form messages and send them out to all neighbouring nodes. The vote messages that are coming from different nodes carry different weights, which are summed up at a receiving node to conclude whether that node can become a cluster head. A node with more neighbours and higher residual energy than others is more likely to accumulate more votes and get a higher weight.

In LEACH [42], each node calculates its chance of becoming a cluster head according to a probability function. The function takes as inputs the estimated optimal number of clusters (energy consumption for communication is involved here) and the total number of nodes. The node with the highest probability value announces itself as the cluster head, and all other nodes that do not elect themselves select their nearest cluster head to join. LEACH distributes the load between the nodes by restricting a node to becoming a cluster head only once in one round, hence preserving energy.

HEED [43] is similar to LEACH in that it periodically calls on the clustering algorithm to elect a new cluster head. However, in HEED, each node has a probability value that determines the likelihood of its becoming a cluster head considering mainly its residual energy. Sensors that are not covered by cluster heads double their probability value until a value of 1 is reached. This process iterates until all nodes are either marked as cluster heads or as regular nodes belonging to a cluster head.

WCA [44] and DCA [45] incorporate more parameters inferred from internal node's properties, which are used in determining cluster heads. Specifically, they suggest considering a node's mobility, degree, residual energy, and distance to neighbouring nodes as inputs to the weighing function. Each contributing property is weighted by a factor to make the algorithm flexible according to the application requirements.

Kulik et al., in [40], develop a protocol, called Sensor Protocols for Information via Negotiation (SPIN), to solve the three broadcast deficiencies. They are Implosion (reception of multiple copies of the same message), Overlap (nodes geographically covering overlapping areas), and Resource Blindness (nodes being ignorant of the energy residual). With SPINS, the number of messages propagated in a sensor network is lessened. The protocol concept is based on negotiation among nodes before they transmit data. The researchers propose a three-step handshake protocol (ADV-REQ-DATA). In the first step, a node advertises through an ADV message that it has new data for transmission. Data is uniquely described by a descriptor, referred to as meta-data. The receiving nodes check the ADV message to see whether they have already received or requested the advertised data. If not, a request response (REQ) is sent back. Data messages are finally disseminated.

In [46], Hong et al. develop an SQL-like language, called Tiny Aggregation (TAG). TAG assumes a sensor network of hierarchical setup. Queries are pushed down to the network to reach a certain area or certain nodes that are declared in the query message. In response to the query and according to its attributes (type of aggregation function, area of interest, and time interval between readings), sensors send their readings to parents, where aggregation takes place. Data is forwarded up the tree and is aggregated at every parent until it reaches the final destination, the base station. The authors give five examples of such queries: COUNT, MIN, MAX, SUM, and AVERAGE. However, they argue that the list can be extended to satisfy different applications.

Directed diffusion is another data-gathering paradigm, proposed by Siva et al. in [39]. The proposed protocol is based on a network of nodes that coordinate among one another to provide collectively sensed information in accordance with operator interest. In directed diffusion, data are named using a set of attribute-value pairs. As the operator initiates interest in a certain sensing task, that interest is disseminated throughout the network to set up gradients. Gradients specify the direction for the gathered data to follow next. Finally, events that match the interest start flowing to the interest originator through one or multiple gradient paths. To save energy, data in this paradigm is aggregated along the path.

More advanced aggregation techniques are found in [47], proposed by Shrivastava et. al. The authors argue that simple types of query such as SUM, COUNT, AVG, and MIN/MAX are not enough in many applications. They propose more complex aggregation functions such as the median, the most frequent data value (consensus value), a histogram of the data distribution, as well as range queries. In their work, they stress that such new aggregation functions are not exact. However, theoretical guarantees on the approximation quality are provided.

3.3.2 Secure Data Aggregation

As per the discussion above, with standard aggregation techniques, it is assumed that sensors are trustworthy, and that having more sensors should provide better results. Unfortunately, those techniques overlook the fact that some sensors may become Byzantine, which will render the network useless.

For that reason, secure data aggregation techniques become vital for the success of sensor network.

Hu. et. al., in [48], address the problem of how to enable secure information aggregation. They consider a network consisting of a powerful base station and a large number of simple sensors. Sensors are spread out enough to allow multi-hop communication between regular sensors and the base station. Each sensor possesses a unique key and shares it with the base station, allowing symmetric cryptographic primitives to be implemented.

The researchers basically propose the use of μTESLA protocol for security [31]. In μTESLA, asymmetry is achieved through delayed disclosure of secret keys. Similarly here, a node is able to verify the child's data validity (through MACs) only after the key is revealed by the base station. The point is that, in the worst case, a node can only lie about its own reading, not another's.

This protocol scales to a hierarchal network, but it is costly in terms of the delay that results from the waiting time before a key is disclosed when data verification takes place at a node. The long waiting time makes it more difficult to implement this protocol especially in large networks or those that require frequent readings. The other problem is that this protocol may fail to ensure data integrity if two nodes in the hierarchy, parent and child, are compromised.

Perrig et. al., in [64], discuss secure aggregation in the sense that a user accepts the data with high probability if the aggregated result is within a desired boundary; otherwise the user detects cheating with high probability and therefore rejects the result if it is outside the boundary.

The researchers start with the assumption that a compromised node, being under full control of the attacker, can commit any sort of harm, including denial-of-service. However, they focus mainly on stealthy attacks, in which the attacker's goal is to make the user accept false aggregate results. The type of network they deal with is assumed to have a large number of simple sensors and one powerful node that does the aggregation task.

To prevent stealthy attacks, they propose a three-stage protocol, named Aggregate-Commit-Prove. In the first stage, aggregation takes place. The aggregator collects raw data from the sensors and computes the aggregation result. Authentication is achieved at this level as each node possesses a key that is shared with the aggregator so as to prevent impersonation.

In the second stage, Commit, the aggregator is responsible for committing to the collected data, that is, to ensure that the aggregator actually uses the data collected from the sensors. To achieve this, they suggest a Merkle Hash-Tree construction [49], in which the aggregator calculates a hash value for every sensor's measurement. The tree is built using those hash values by hashing at each of the tree nodes the concatenation string of the node's children. The root of the tree is called the commitment of the collected data.

In the final stage, Prove, the aggregator and the home server establish a hand-shake protocol to prove that the aggregation results are correct.

In terms of cryptography and communication, symmetric cryptographic functions are used here for authentication, integrity, and encryption. Each sensor has a unique identification and shares a separate key with the home server and with the aggregator. In fact, they store only master keys, while the sensors produce communication keys by hashing the node id with the master key, i.e., $HMAC_K$(node id).

This solution focuses on obtaining the best approximation of the aggregation results with the existence of Byzantine nodes rather than on identifying the misbehaving nodes. In fact, with this setup, compromising a single node will make key material for all the nodes available to an attacker. Furthermore, compromising only a single node in addition to the aggregator enables the attacker to make all sorts of attacks, including stealthy attacks, which the SIA protocol is mainly trying to defend against because in such a case, the aggregator can learn the key material of the nodes and thus fake the whole Merkle hash tree.

Another problem is related to scalability. As the number of regular sensors increases, the computations involved, such as the construction of the Merkle hash-tree, become more challenging in terms of processing power.

In [50], Deng et al. propose having witness nodes to achieve secure data aggregation, that is, whether the data fusion to the base station is valid. In their scheme, each sensor, including the witness nodes, shares a unique key with the base station. The idea is that witness nodes, being in the same radio range as the aggregator, also perform data aggregation and compute the corresponding MACs using their keys. Then they send those MACs to the aggregator, which, in turn, forwards them to the base station. Then, as the base station has received the aggregate result from the aggregator along with a number of corresponding MACs, it is better able to verify the correctness of the fused data.

Finally, Wu et al., in [51], develop a secure aggregation mechanism that lets child nodes monitor father nodes in a tree topological setup to report any cheating. A child node is assumed to be able to listen to all messages sent from its siblings to its father node. To do so, child nodes build an aggregation tree, called Secure Aggregation Tree (SAT), by propagation of invitation messages. A node that receives an invitation message marks the sender as the father and then broadcasts similar invitation messages to cover all the nodes in the area. To detect cheating, each child node, acting like a watchdog, keeps observing all the messages sent to its father and the aggregate messages sent from the father to its grandfather as well. If a node's father sends out a value that is significantly different from a correct aggregate value, the node will raise a weighted alert, indicating the severity of the suspected cheating. As a result, when the number of alerts from different nodes exceeds a certain threshold value, SAT is rebuilt so that a misbehaving father is excluded. The interesting feature in this technique is the light use of cryptography; for example, only alert messages are authenticated. On the other hand, the shortcoming is that leaf nodes cannot be checked.

In conclusion, the unattended nature of sensor networks makes them prone to many types of attacks, including some unique ones, such as insider attacks. When a node is compromised, an adversary can easily inject false data into the network, resulting in severe consequences: distorted data integrity, depleted battery power, and degraded bandwidth utilization. However, designing a sensor network that is able to detect false data injected by compromised nodes is a great research challenge because compromised nodes act as part of the network by possessing cryptographic keys. When aggregation is required, the problem is compounded because nodes legitimately can alter the data for the sake of decreasing redundancy. Consequently, nodes cannot differentiate between cheating and aggregation. The next chapter discusses the method proposed for detecting injected false data; it shows how to filter out false data, how to punish responsible nodes for misbehaviour, and how to assign trust values to sensors.

CHAPTER 4 SECURE DATA AGGREGATION

This chapter presents the proposed secure data aggregation approach that is robust to Byzantine nodes. This approach is able to detect Byzantine sensors, assign trust values to each sensor, and determine when to apply cryptography. The chapter is organized as follows:

1) System Model and Assumptions: This section explains some basic assumptions about the sensor network setup. Furthermore, it states the thesis goal from a security point of view.
2) Solution Framework: This part presents a strategy to Byzantine behaviour detection. It demonstrates how false data is discovered at the aggregator node and how the proposed system assigns trust values to sensors accordingly.
3) Performance Analysis: This section evaluates the performance of the proposed secure aggregation method. Performance evaluation involves simulation results, complexity analysis, and energy savings.

4.1 SYSTEM MODEL AND ASSUMPTIONS

Consider the scenario of a network of wireless sensors deployed in a certain area to perform measurements. Because the sensors are assumed to be simple, low in power consumption, and short in communication range, there exist intermediate nodes with relatively higher processing capabilities called aggregators. Upon a query from the home server, sensors perform their measurements and report to the aggregator, which, in turn, performs some processing and eventually sends the result to the home server (Figure 4-1).

The goal is to detect and avoid possible Byzantine nodes, those which report wrong information to the aggregator. Node assessment takes place at the aggregator node, which is assumed to be a trustworthy node, similar to the home server.

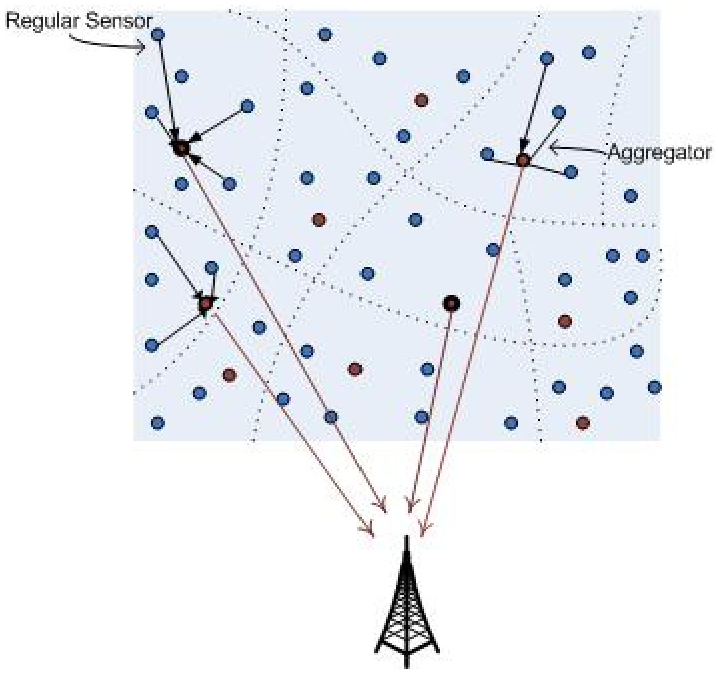


Figure 4-1 Aggregation Network

4.1.1 Notations

The following notations are used in the description of the secure aggregation procedure.

- n is the number of nodes.
- $S_1, \ldots\ldots, S_n$ are the regular nodes.

- H is the home server or base station.
- A is the aggregator.
- X_i is the value reported by S_i.
- $K_{H,Si}$ is the shared key between sensor i and the home server.
- $K_{A,Si}$ is the shared key between the aggregator and home server.
- $m_1|m_2$ denotes the concatenation of two messages, m_1 and m_2.
- $E(K,m)$ refers to the encryption of message m using key K.
- $MAC(K,m)$ is the message authentication code of message m with key K.
- $Si \rightarrow A$: m means a sensor Si sends a message m to the aggregator A. In a similar manner, $A \rightarrow S_i$:m and $H \rightarrow S_i$:m are defined.
- Agg is the aggregate result that the aggregator node produces.

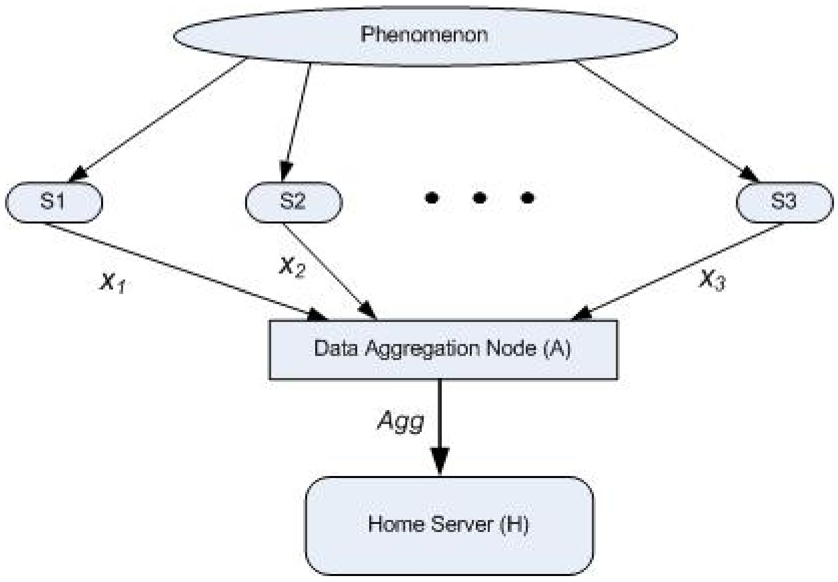


Figure 4-2 – Block Diagram of a WSN Aggregation Network

4.1.2 Security Goal and Assumptions

The goal is to protect the wireless sensor network from Byzantine nodes. The main threat considered here is that from malicious data, which can propagate in a network in many ways, for instance, through node compromise, or by physically spoofing the sensing element, e.g., heating it up. Our goal is to protect the wireless sensor network from Byzantine nodes, identify them, and filter them out. Byzantine nodes, by definition, include compromised nodes that participate as part of the network. A compromised node is completely under the control of an adversary and is assumed to act skilfully. In this context, a skilful attack refers to the ability of an adversary to lie about its measurement at times and simply to be honest at other times. An adversary's goal, in attacking nodes, is to severely deviate the final aggregated result from the true value.

In this work, The adversary is assumed to be polynomially bounded, which leads to the assumption that compromising a sensor is not an easy task. Therefore, only some of the sensors can fall under an attack – say 30% of the regular nodes. That is considered a safe number.

Further, corrupted sensors may collude as they can be controlled by the same attacker. Collusion, in fact, is not very likely to happen because sensors are synchronized with the aggregator so that they report their measurements at the same time; however, this assumption is made to strengthen the notion of security.

The aggregators, in this setup, are regarded as trusted nodes. The number of aggregators is far less than the number of regular nodes. Hence, more can be spent on them to enhance their security abilities, for example, enclose them in high levels of tamper-resistant packaging or mount them in places where high security standards can be guaranteed. In addition to that, aggregators have heavier duties than other regular sensors because they act as centres to process data coming from other sensors. If aggregators die, then the whole network dies. Therefore, more processing power and more energy are required for aggregating nodes.

4.2 THE SECURE DATA AGGREGATION PROTOCOL

4.2.1 Key Setup

Initially, cryptographic tools and secret keys are installed on all sensors; however, the use of them is avoided until misbehaviour is detected.

As concluded in the discussion in Chapter 3 on symmetric and asymmetric cryptography, the former is more appealing in such a resource-constrained environment. Therefore, each node possesses two keys, preloaded before deployment. One is shared with the aggregator and the other one is shared with the home server. Regarding identification, every node owns a unique ID.

For efficient use of keys and storage space, the home server and the aggregator do not store separate keys for every single sensor. Instead, each of them, the home server and the aggregator, shares unique keys with the regular nodes in the form of a single master key. That is, the home server stores the master key K_H, and each node stores MAC(K_H,node ID). Similarly the aggregator stores the master key K_A, and every nodes stores MAC(K_A,node ID). Although nodes derive their unique keys from the same master key, a compromise of one node does not spread to others. In other words, learning the master key from the node key is not possible because of the one-way property that the MAC function holds. With this technique, the home server and the aggregator are able to compute the keys they share with the regular nodes by using the master key to hash the node ID. In the same manner, the hashing technique is further extended to derive more computationally-independent keys for encryption and data authentication. Figure 4-3 shows our procedure for key derivation.

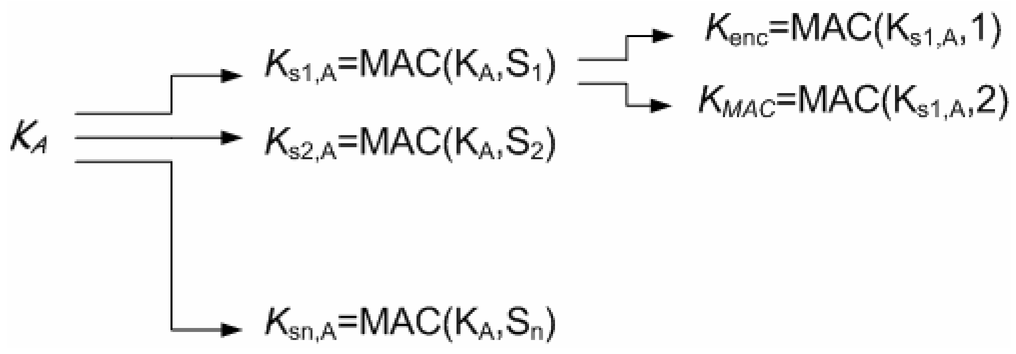


Figure 4-3 Deriving keys from the master secret key

If a new node is added to the system, the corresponding key is added in the system, to the home server, and to the aggregator. However, for security reasons, the aggregator key can be changed and disseminated to all the nodes when needed. Nodes communicate with the home server through the aggregator. In parts of our communication protocol, the nodes exchange special information with the home server using the home server node keys. Even though communication takes place through the aggregator, obviously the latter is not able to reveal such information. Communication between neighbouring nodes is not part of the current set-up, so pair-wise key sharing is not required. In certain applications pair-wise communication is involved. Protocols such as the ones discussed in [14,52,53], and [54] can be implemented for that purpose.

4.2.2 Protocol Overview

Below is an abstract overview of the message exchange that takes place between the three main components – the home server, aggregator, and regular sensors. The steps demonstrated here are elaborated on in later sections.

The home server (or base station) initiates the sensing process by sending out a broadcast message to certain sensors that are located in the area of interest. Through the broadcast, nodes, including suspicious (less trustworthy) nodes themselves, also learn which sensors are less trustworthy.

1. Regular sensors then report back measurements through the aggregator as messages cannot reach to the home server directly.
2. The aggregator processes the received measurements. In addition to the aggregation functionality provided, it identifies outlying sensors through an outlier detection algorithm. At the same time, it assigns trustworthiness values to every sensor to keep track of the sensors' performance. The aggregation result along with Byzantine sensor IDs are reported to the home server.
3. According to the trust values, certain decisions are made, e.g., to force nodes to run certain cryptographic functions or to eliminate a node completely.

Steps 3 and 4 can been seen in the following sections in three main points:

- ***Outlier Detection:*** the procedure used to discover misbehaving nodes.

- ***Trust Establishment:*** a mathematical model to derive an appropriate trust measure.
- ***Communication Messages:*** a demonstration of the ways different messages are formatted showing how cryptographic functions, when needed, fit here.

4.2.3 OUTLIER DETECTION

Intuitively, without malicious behaviour and according to the central limit theorem, which states that the sum of many independent and identically-distributed random variables with finite variances will tend to follow normal distribution, measurements reported by sensors comply with this definition, so they appear to the aggregator to follow normal distribution. The existence of one sort of attack or another makes some measurements differ from the rest.

Luckily, those malicious attacks can be treated as outliers, which are a well-studied problem in statistics. From [55], the problem of outliers is defined as "an observation that deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism."

Many outlier detection schemes have been introduced in the literature. Generalized Extreme Studentized Deviate (GESD) [56] is among them, and has proved to perform very well under various conditions, as argued in [55]. Many papers such as [57],[58], use GESD and conclude that it outperforms other detection procedures. The following section explains how GESD many-outlier detection works followed by a section that demonstrates how this detection algorithm would fit with our problem. Recall that outliers for us represent a possible attack.

4.2.3.1 Generalized Extreme Studentized Deviate Algorithm (GESD)

GESD is a statistical procedure introduced by B. Rosner [56] and is based on the ESD test, which detects one outlier at a time. The following section details how ESD works.

<u>ESD Test</u>

Given a set, X, of n data samples where $X \in \{x_1, x_2, \ldots\ldots, x_n\}$, the mean of X is denoted as x', and the standard deviation of X is denoted as s.

Let T_i (the corresponding T-value or xi) = $|x_i - x'| / s$, where i = 1,2, … ,n.

Let x_j be the observation that leads to the largest T-value. Then, x_j is an outlier if Tj exceeds a tabled critical value λ. Otherwise if Tj does not exceed the critical value λ, it may be concluded that the test finds no outliers.

If the test finds an outlier, that outlier is filtered out, and test continues the same process with the remaining (n-1) points to look for further outliers.

<u>GESD Test</u>

GESD is a modified version of the ESD test and can identify multiple outliers in the data set. Before analysis, two parameters must be specified, the probability of incorrectly declaring the existence of outliers when no outliers exist (sensitivity of the process), λ_I, and an upper bound on the estimate number of outliers, Nu.

Formally, λ_i is the two-sided 100α percent critical value and can be calculated from the following formula.

$$\lambda_i = \frac{t_{n-i-1,p}(n-i)}{\sqrt{(n-i-1+t^2_{n-i-1,p})(n-i+1)}} \qquad \ldots(1)$$

where $i = 1, \ldots, N_u$

$T_{v,p}$ is the 100*p percentage point from the t distribution with v degrees of freedom, and $p = 1 - [\alpha / 2(n-I+1)]$.

Given α, n, and N_u, the critical values λ_i, where $i = 1, \ldots, N_u$, can be calculated beforehand. In other words, the probability is adjusted so that the number of warnings about the outliers is kept to an acceptable level. The upper bound must be less than half of the number of data samples, i.e., $Nu \leq 0.5\ (n-1)$, since the number of outliers cannot form the majority, as that would negate the original definition of outliers.

Algorithm: Identifying outliers with GESD

```
Input: $N_u$, X, $\lambda$
    Begin loop (I=1,2,3,…,$N_u$)
        Compute average x' of elements in set X
        Compute standard deviation s of elements in set X
        If s = 0 continue (next i)
        Find i^th extreme studentized deviate $T_i$
        Calculate i^th critical value $\lambda_i$
        If $T_i > \lambda_i$ then outlier
            Move element $x_i$ from set X to outlier list
    End loop
    Return outlier list, non-outlier list
    Calculate robust estimate of the mean ($x'_{robust}$) from set of non-outlier list
```

Calculate robust estimate of the standard deviation (s_{robust}) from set of non-outlier list

4.2.3.2 Choosing Nu (Maximum Estimated Number of Byzantine Nodes)

It is important that to choose Nu carefully as it plays a primary role in GESD. Too small Nu could mean some Byzantine nodes would not be reported. On the the hand, too large Nu would make GESD waste time checking nodes that are in reality good ones (honest).

According to Becher et al. [60], launching attacks based on physical node capture is not easy and requires expert knowledge, costly equipment, and other resources. Removal of a sensor from a network can be noticed by its neighbours and the node itself.

Wagner, in [61], argues also that the adversary's capabilities are not unlimited. Some cost or luck will be required for each node that the adversary wishes to compromise. Therefore, he suggests the assumption that an adversary can compromise only a limited number of nodes but not half of the network. A network that is robust to one or two or three compromised nodes indicates a network that is in good shape.

Accordingly, this work does not consider an extreme case, half of the nodes, nor does it assume that only a very few of the nodes are compromised. The percentage 30% is a reasonable assumption. For example, with a network of 20 sensors, an assumption of up to six possible compromised nodes is sensible.

4.2.3.3 Choosing α

Rosner, in [56], demonstrates the GESD algorithm by assuming three different values for α: 0.05, 0.01, and 0.005. The symbol α represents the percentage of the tail portion's distribution that is far from the mean. The greater the percentage value, the greater the number of points that can be included. From the security point of view, "α=0.05" (representing the strictest security) is preferred (Figure 4-4).

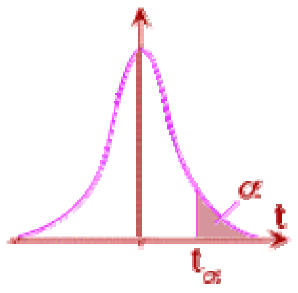


Figure 4-4 Student's t Distribution

4.2.3.4 The Role of GESD in Evaluating Sensors

GESD is the first step towards identifying Byzantine sensors. When information {x1, …, xn} is reported to the aggregator from different sensors in the network {S1,…, Sn}, some of the measurements deviate from the true value as a reflection of Byzantine behaviour. Regardless of the aggregation function, the aggregator analyzes the received readings and points out the extreme ones (sensor IDs). Accordingly, trust values are assigned to sensors (next section).

4.2.4 Trust Establishment

This section describes how trust measurement is assigned to individual sensors. First of all, an important question is where should sensors assessment take place? Possibility number one is to let the aggregator transfer the aggregation result along with the sensor IDs of outlying sensors to the home server, which is where the trust value is calculated and updated and a decision is taken accordingly as to whether to force some of the nodes to use cryptography for future communication or even to eliminate them from the network. Possibility number two is to let the aggregator do the complete job – assign and update trust values and decides on the appropriate punishment for deviant nodes. The first setup seems appealing because it allows future, more general, expansion of the secure aggregation protocol by involving the aggregator itself in the assessment. This would suggest that the aggregator is not completely trusted, which suits some sensor network scenarios. The second setup suits some other scenarios as well, in which aggregators are considered resource-enhanced stations, so every aggregator assesses its directly reporting nodes (4.1.2). This setup has the advantage of the ability to apply the security scheme on a hierarchical aggregation network. For now let us assume that the aggregator is the one that computes trust and makes the decisions.

After calculating the aggregation result and identifying outlying sensors, the aggregator assigns a trust value to each regular sensor and keeps updating this value according to the sensors' performance. The aggregator makes the decision as to when a sensor should use cryptography and when it should be eliminated from the network. Below is a demonstration of how trust measure is derived.

To begin, let us first find a suitable definition of trust. There are several definitions of trust in the literature. According to Linag and Shi, in [62], It is "*the subjective probability by which an individual A expects that another individual B performs a given action as good as expected.*" That means "A" would be applying a probabilistic measure as to how likely 'B' is to do certain tasks properly. A metric, hence, is needed to measure that possibility. Sun et al., in [59], take us steps ahead towards information theoretic concepts by saying "*trust is a measure of uncertainty";* as such, trust value can be measured by entropy.

Therefore, trust is a function of entropy. Applying that definition to the matter of aggregation in sensor networks, trust determines how uncertain the aggregator is about the other nodes as to whether they would report good and honest measurements or not. In other words, the home server observes the performance of the nodes reporting to it, then derives and associates a trustworthiness value with each one of them.

To formulate that, Sun et al., in [59], in studying the performance of ad hoc network routing protocols, propose a continuous function to determine trust. The following shows how they derived and used their trust model in their scenario, followed by a demonstration of how the equations and definitions are manipulated from its original context so that they adapt to our case.

In an Ad Hoc network, packets get routed through multiple nodes. Assume that node "A," called the subject, sends a set of packets to some destination through an intermediate node "B," called the agent. Equation 1 calculates how much trust the subject A can have in the agent B to do a certain action, which is forwarding packets in this case.

$$T\{\text{subject; agent; action}\} = \begin{cases} H(p)-1 & , 0 \le p < 0.5 \\ 1-H(p) & , 0.5 \le p < 1 \end{cases} \quad \ldots(1)$$

$$H(p) = -p \log_2(p) - (1-p)\log_2(1-p) \quad \ldots(2)$$

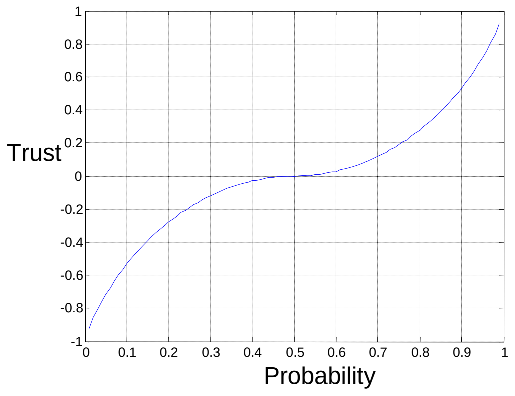


Figure 4-5 - Trust as a function of entropy

where

- T{subject; agent; action} (also written T_{AB}) is the trust value that the subject A holds for the agent B.
- H(*p*) is Shannon entropy.
- *p*: P{subject; agent; action} is the probability estimate that the subject holds for the agents as to how likely the latter is to perform the action next time.

From Figure 4-5, when probability, p, equals 1, the aggregator is certain about the honesty of the node, and trust is at its maximum value, which is 1. On the other hand, a probability of 0 means that the aggregator is certain about the dishonesty of the node, and hence, trust is at its lowest value, which is -1. Moreover, when the probability is ½, trust becomes zero, which means that the aggregator has no trust in the node.

It is important to show how Sun et al. calculate the probability. According to their routing scenario, each mobile node holds a trust value for each of the other nodes in the network as per the following: a node calculates the trust of forwarding packets, that is, how trustworthy an agent is in forwarding packets. The value is derived from the number of successfully delivered packets, *K,* and the total number of initially sent packets, *N,* according to Equation 3 or Equation 4 (taking into account the time of observation). If a node holds no trust value for a certain destination, then trust is established by getting recommendation values from other nodes in the network. Figure 4-6 depicts a typical setup of mobile nodes. For example, A wants to send packets through D, so it first updates the trust value associated with D from its past observations and from other nodes' recommendations.

$$P\{\text{subject: agent, action}\} = \frac{1+k}{2+N} \qquad \ldots(3)$$

$$P\{\text{subject: agent, action}\} = \frac{1+\sum_{j=1}^{I}\beta^{t_c-t_j}k_j}{2+\sum_{j=1}^{I}\beta^{t_c-t_j}N_j} \qquad \ldots(4)$$

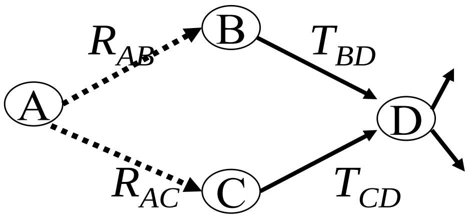


Figure 4-6 - A typical example of trust propagation

Now, moving back to aggregation in sensor networks, The above formulations are manipulated so that they fit into the problem and become part of the overall solution. For that, let us assume that the aggregator represents the subject while the regular nodes represent the agents. The action is the measurement provided every time by each node. Therefore, the scenario changes thus: the aggregator identifies sensors whose measurements are classified as bad to the home server. The home server updates k, the number of good measurements reported by a certain sensor, and N, the total number of readings.

Equations 3 and 4 can be used in calculating the trust value. However, they do not completely catch all the issues connected with aggregation, for example the severity of punishment when a node is suspected of misbehaviour or how this misbehaviour is linked with the environment in terms of the probability of nodes being attacked and compromised. If a sensor network is deployed in an area that requires higher security measures, then calculation of trust should reflect that. For example, trust values drop fast but build relatively slowly. For that reason a punishment factor is introduced (equations below).

$$P\,(\text{Trust}) = \begin{cases} \text{Good measurement} \quad \dfrac{1+k}{2+N} & k=k+1\text{ , }N=N+1 \\[2em] \text{Bad measurement} \quad \dfrac{1+\beta\ k}{2+N+\Delta S} & k=k+0\text{, }N=N+1 \end{cases} \quad \dots (5)$$

where K is the Count of good measurements

N is the Total number of measurements

ΔS is the Number of standard deviations from mean ($\Delta S = \frac{x_i - x'_{robust}}{S_{robust}}$)

X'_{robust}, S_{robust} are calculated from algorithm 1.

β is the Punishment factor ($\beta = (1\text{-}0.6P_A)$) …(6)

P_A is the probability of attack

Equations 1 and 2 then follow to obtain the trust value in terms of entropy.

$H(p) = -p\ log_2(p) - (1\text{-}p)\ log_2(1\text{-}p)$

$$T\{\text{subject; agent; action}\} = \begin{cases} H(p)-1 & , 0 \le p < 0.5 \\ 1-H(p) & , 0.5 \le p < 1 \end{cases}$$

4.2.4.1 Threshold and Probability of Attack

Trust ranges between –1 and 1 (Equation 1) with an initial value of zero (pr=0.5). In this scheme, different threshold values can be introduced to make the system more dynamic and more flexible to designer discretion. For example, with multiple threshold values, each of which defines a required level of security (e.g., a longer key, a stronger cryptographic function, or other methods), as a sensor's trust value drops below a certain threshold value, that level of security is demanded. The worst case occurs when the trust value is lower than the lowest threshold value, resulting in removing the node from the network. For simplicity, two threshold values are maintained: the first one defines when to enable cryptography and the second one defines when to remove a node.

In Equation 5, β (punishment factor) is plugged in to make the equation sensitive to bad behaviour. Probability of attack P_A(β=1 - 0.6 P_A) defines how likely a node is to be compromised. By setting P_A to a certain value, sensitivity increases or decreases accordingly.

Figure 4-7 demonstrates a hypothetical scenario of a node with different values for β. Trust keeps increasing for that node until it reports a measurement that does not comply with the consensus. At the measurement number 50, the trust drops abruptly. Probability of attack P_A determines the severity of trust decrease. For example, for a P_A as high as 0.9, trust drops below the threshold value in one shot. Next, good measurements are reported so that trust increases; however, the increase is slow. For the sake of security, trust is assumed to drop quickly and build slowly.

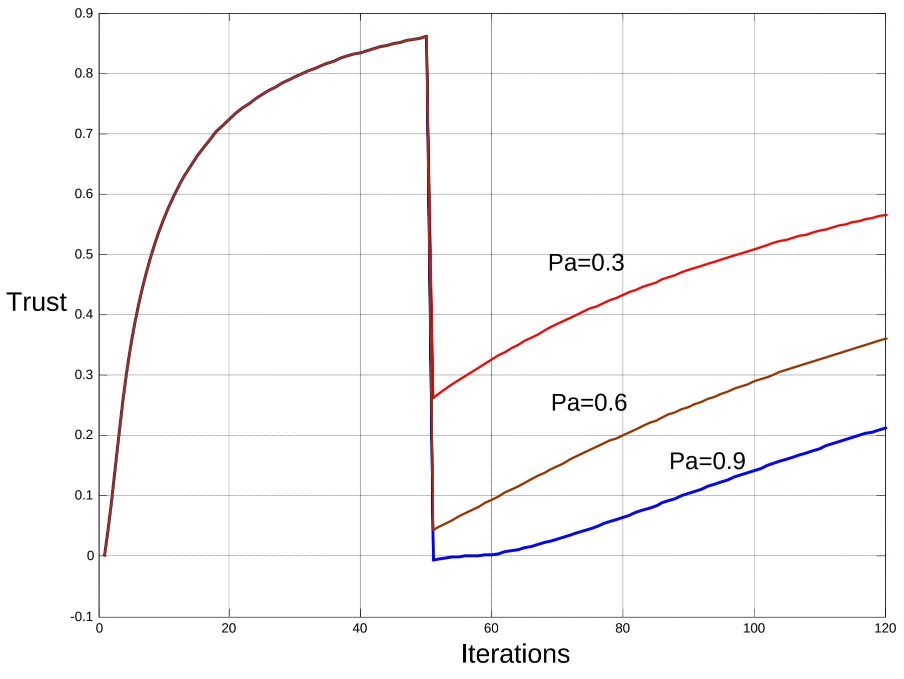


Figure 4-7 Trust Function- Hypothetical Example.

4.2.5 Communication Messages

This section introduces the packet format for the different communication messages that are sent among the three node types: regular sensors, aggregator, and the home server. The two protocols, SNEP (for data confidentiality, authentication, integrity, and freshness) and μTESLA (for authenticated broadcast queries), are borrowed for the benefit of our secure aggregation approach with a slight modification. Chapter 2 introduces these two protocols, but for more details, refer to [14].

4.2.5.1 Broadcast (Query Dissemination)

In the process of query dissemination from the home server to the network, sensors need to learn mainly about the aggregation function, f_{AGG}, they are to provide. Besides the aggregation function, a randomly generated number is attached to ensure freshness (prevent replaying). Several configuration parameters precede the query message to supply the receivers with the information required for authentication. Such information includes time duration, a time stamp (T_s) to provide time synchronization, a randomly generated number N_H to provide freshness, and MAC tag to achieve authentication. Other fields related to the key disclosure include starting time Ti, time interval T_{int}, and the disclosure delay d.

$$H \rightarrow S*: T_s|K_i|T_i|T_{int}|d, MAC\ (K_{H,Si}, N_H|T_s|K_i|T_i|T_{int}|d)$$

A number of query messages, following the previous setup message, can be sent out before disclosure of the secret key K_j. A query message may look like the following message.

$$H \rightarrow S_*: Fagg, N_H, MAC(K_j, F_{agg}|N_H)$$

4.2.5.2 Sensors to Aggregator

Opposite to query dissemination, data aggregation flows starting from the regular nodes and ending at the home server. Again, in a trusted environment, sensors send simple packets that carry their IDs and readings to the aggregator. However, when cryptography is enforced, nodes encrypt the packets (if confidentiality is required) and add MAC tags to them. The secret key used here is the one shared between the node and the aggregator. The following shows the packet that a regular node, S_i, sends to the aggregator.

$$S_i \rightarrow A: S_i, E(K_{Si,A}, X_i \mid N_H) \mid MAC(K_{Si,A}, S_i| X_i| N_H),$$

where X_i is the data reported by node S_i, and N_H is a random number to identify the query and to prevent replay attacks.

4.2.5.3 Aggregator to Home Server

When the aggregator receives all the packets from all sensors, it first filters out outlier readings, assign trust values to each sensor, and then performs the aggregation. Before cryptography is switched on, the aggregator reports to the home server the aggregate result and the IDs of those sensors whose trustworthiness drops below the set threshold in a packet that looks like the following one:

$$A \rightarrow H: A, Agg, Byzantine\_count, Byzantine\_IDs, N_H$$

With encryption and authentication enabled, the aggregator first decrypts the data it receives from the regular sensors using keys it shares with them. Then it uses its shared key with the home server to form the following packet.

$A \rightarrow H$: A, $E(K_{Aenc,H}$, $Agg|\ Byzantine\_count|\ Byzantine\_IDs|\ N_H)|$

$MAC(K_{Amac,H}$, $A|\ Agg|Byzantine\_count|\ Byzantine\_IDs|\ N_H)$,

where Byzantine refers to the nodes that have low trust values less than a threshold. NH is the same as the one used in the initial query message and in the packets reported by the regular nodes.

4.3 Secure Hierarchical Aggregation

If the sensor network is too large, which is common, then multiple aggregators, usually cooperating, are required to handle the entire network. Functions such as AVERAGE, MIN, and MAX do support hierarchical aggregation. That is, every aggregator performs the aggregation function on a subset of the nodes in the sensor network. The results are collectively sent to other aggregators for computing the same aggregation function again, and eventually deliver the final result to the home server (Figure 4-8).

If hierarchical aggregation is required, my proposed approach is still applicable (Byzantine detection and trust assignment), but with a slight modification in the message format. Aggregators can still send messages to other aggregators (e.g. cluster heads) and, depending on the type of application, the first aggregator can decide whether the second one should perform aggregation or not. For that, hierarchical aggregation can be introduced into two different techniques:

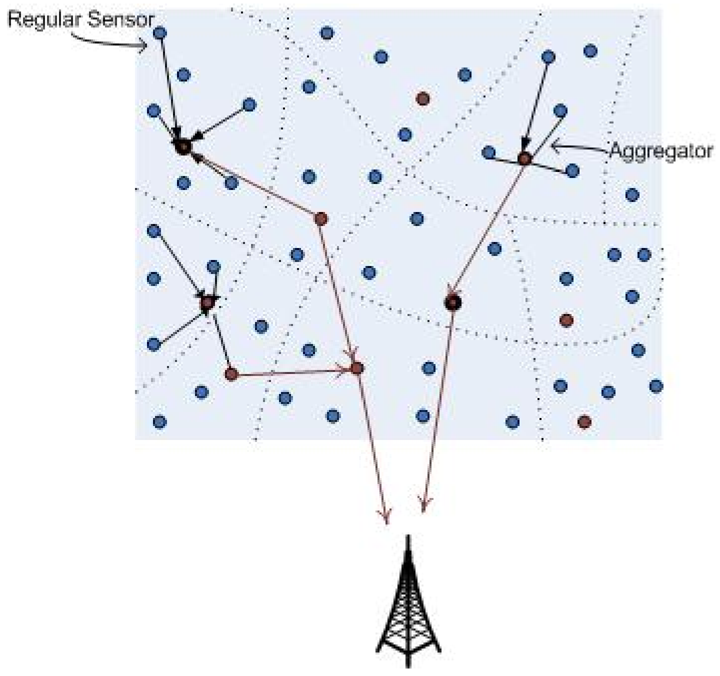


Figure 4-8 Hierarchical Aggregation Network

In technique number one, each aggregator processes only the information received from its direct sensor group and then sends that aggregate result straight up to the home server through other aggregators (e.g., inter-cluster communication). However, the latter forwards the aggregate message without performing any more aggregation. In this technique, pair-wise keys shared between one aggregator in the hierarchy (child) and the next one (parent) are not required. However, the parent aggregator needs to learn that it has to forward the aggregate packet intact. An extra flag bit (aggregate bit) is added in the message for every aggregator on the way to the home server to check. A bit of "0" indicates that no further aggregation is needed. The following shows the message format that an aggregator A_{child} sends to another aggregator A_{parent}. The first message is without cryptography while the second one encrypts and authenticates.

$A_{child} \rightarrow A_{parent}$: A_{child}, 0, Agg, Byzantine_count,Byzantine_IDs, N_H

$A_{child} \rightarrow A_{parent}$: A_{child},0, E($K_{Aenc,H}$, Agg| Byzantine_count| Byzantine_IDs| N_H)| MAC($K_{Amac,H}$, 0|A_{child}| Agg|Byzantine_count| Byzantine_IDs| N_H)

In technique number two, a global aggregation is required. That is, every aggregator performs aggregation on its group sensors as well as on the results received from other aggregators. Our approach adapts to this technique through a modification in the packet format, in the three fields: aggregate bit, count value, and pair-wise key sharing if cryptography is on. An aggregate bit of "1" indicates to the aggregator that further aggregation is required. The count value is also necessary to show how many aggregators have already contributed to the aggregated data before further aggregation can be applied. Regarding the key sharing between two aggregators in the hierarchy, the second one uses the key to decrypt the packet before carrying on with aggregation. Encryption here provides confidentiality and also authentication because keys are shared only between the two aggregator nodes. Several key management protocols have been suggested in the literature. As a matter of fact, the best way to bootstrap secure connections is public-key cryptography for symmetric setup. However, as explained in Chapter 2, public-key cryptography is not a viable solution in such a resource-constrained environment. Therefore SPINS [31] node-to-node key sharing method is used here for key setup. Because the home server is regarded as a trusted machine, distributing keys between two communicating aggregators can be delegated to the home server.

Assume that the child aggregator A_c wants to establish a secure connection with the parent aggregator A_p to deliver the aggregate data. Both aggregators share no keys, but both share keys with the home server (K_{Ac} and K_{Ap}) and trust it. The home server acts as a key distribution centre (KDC) to distribute the symmetric key $K_{Ac,Ap}$ to both parties through the following protocol.

1: $A_c \rightarrow A_p$: N_{Ac}, A_c

2: Ap $\rightarrow$ H: N_{Ac}, N_{Ap}, A_c, A_P, MAC(K_{Ap}, N_{Ac}| N_{Ap}| A_c| A_P)

3: H $\rightarrow$ A_c: E(K_{Ac}, $K_{Ac,Ap}$), MAC(K_{Ac}^{mac}, N_{Ac}| A_p| E(K_{Ac}, $K_{Ac,Ap}$))

4: H $\rightarrow$ A_p: E(K_{Ap}, $K_{Ac,Ap}$), MAC(K_{Ap}^{mac}, N_{Ap}| A_c| E(K_{Ap}, $K_{Ac,Ap}$))

where N_{Ac} and N_{ap} are two random numbers to ensure key freshness. $K_{Ac,Ap}$ is the key to be shared between the two communicating aggregators. The keys used for encryption are different from those used in authentication (K_{Ac}^{mac} and K_{Ap}^{mac}). In line 2, MAC is added to prevent DOS attacks as the home server is made to answer only authenticated requests.

The packet submitted in technique two looks like the following:

$A_{child} \rightarrow A_{parent}$: A_{child},1, E($K_{Aenc,H}$, COUNT| Agg| Byzantine_count| Byzantine_IDs| N_H)| MAC($K_{Amac,H}$, 1|COUNT|A_{child}| Agg|Byzantine_count| Byzantine_IDs| N_H)

4.4 Performance Evaluation

This section provides an analysis to the performance of the proposed secure aggregation method. First it discusses the complexity of the algorithm in terms of the computations involved. Next, through simulation, it evaluates the procedure in terms of the three security metrics that are commonly used: successful detection rate, false positive rate, and accuracy improvement rate. Finally, it shows how much energy is expected to be saved.

4.4.1 Algorithm Complexity

GESD is the main component in our method and is run at the aggregators. In GESD, calculating the standard deviation, S, is the most time-consuming operation in terms of the number of multiplications involved.

$$S = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (x_i - \overline{x})^2}$$

Given n and r, the number of multiplications can be calculated.

n + (n-1)+(n-2)+…+(n-r)

In general, GESD time complexity is O(nr).

The number of outlier measurements, r, is assumed to be n/2 at worst. Hence, time complexity is $O(\frac{3}{8} n^2) \approx O(n^2)$.

4.4.2 Simulation Setup and Results

The performance metric used in security analysis is the ability to detect malicious data with the existence of Byzantine nodes. The simulation scenario is set to calculate, specifically, the successful detection and false detection rates. By definition, the successful detection rate is the percentage of Byzantine nodes that can be detected. On the other hand, the false positive rate shows the percentage of the nodes that are reported Byzantine but are not. A third metric is the accuracy improvement rate. As the system excludes malicious data, measurement accuracy improves. Accuracy depends on the aggregation function in question; however, the accuracy improvement can be determined in terms of the deviation from the mean.

In the simulation, a number of sensors are deployed in an area to perform a certain measurement to the same or correlated environmental events. The sensors report contaminated measurements with noise (N(0,1)) to a collection sensor, the aggregator. The scenario is simulated with one aggregator and 25 sensors (S_1-S_{25}), from which up to 7 sensors might become compromised at once (that is, 30% of the total number of sensors). The probability of attack, P_A, is assumed 0.3 (β =0.82).The results are presented below.

The Successful Detection Rate can be calculated as the number of detected Byzantine nodes over the real number of Byzantine nodes in the sensor network. There are hardly any undetected Byzantine nodes because the trust function keeps track of the nodes' performance, and, accordingly, the associated trust value is updated. It is possible that a Byzantine node may operate while staying undetected for a few readings, but eventually, its trust value drops and it is marked Byzantine. As explained above, the trust values drop relatively fast as compared to growing. That makes the trust function fairly smart and thus effective. Figure 4-9 shows an example of three nodes. The red one (S_{25}) is a Byzantine node that tries to hide by reporting good measurements at times and bad measurements at other times. Its trust value, though, drops below the threshold value after a period of time, while S1 and S10 continue to improve their trustworthiness status.

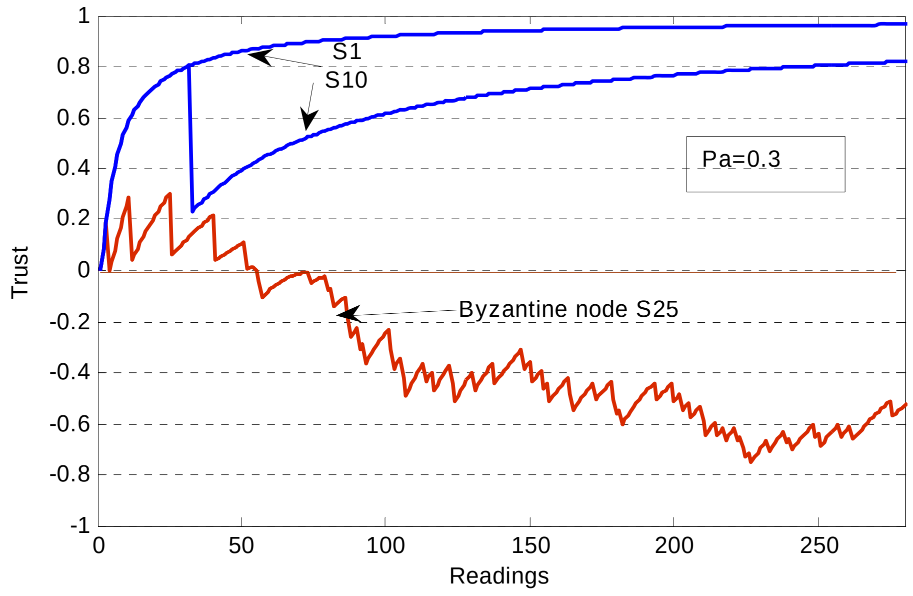


Figure 4-9 Sensor Node Trust (β =0.8)

Figure 4-10 demonstrates the result of trust values for every sensor. With P_A equal to 0.3, Byzantine nodes S_{19}-S_{25} were captured in approximately 45 to 50 readings, a mixture of good and malicious ones.

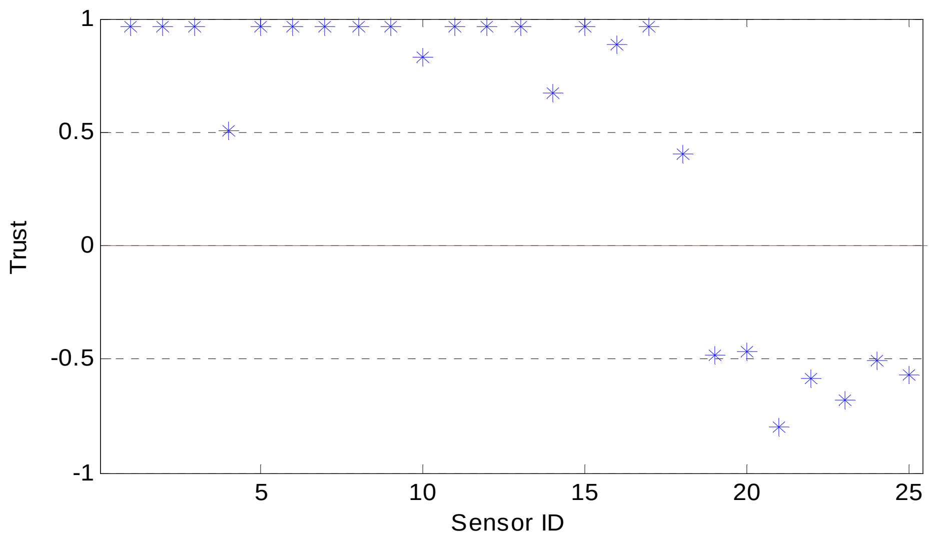


Figure 4-10 Trust Result

Obviously, Byzantine nodes can be captured faster with a higher probability of attack, P_A. Probability of attack defines how risky the environment is in terms of the possibility of attacking the nodes and compromising them. With a higher P_A, the system is more sensitive to malicious data and tends to punish misbehaving nodes faster (Equation 5 and Equation 6). Figure 4-11 shows that the Byzantine node S_{25} was detected in 20 measurements, much faster than the case with Pa=0.3 (Figure 4-9).

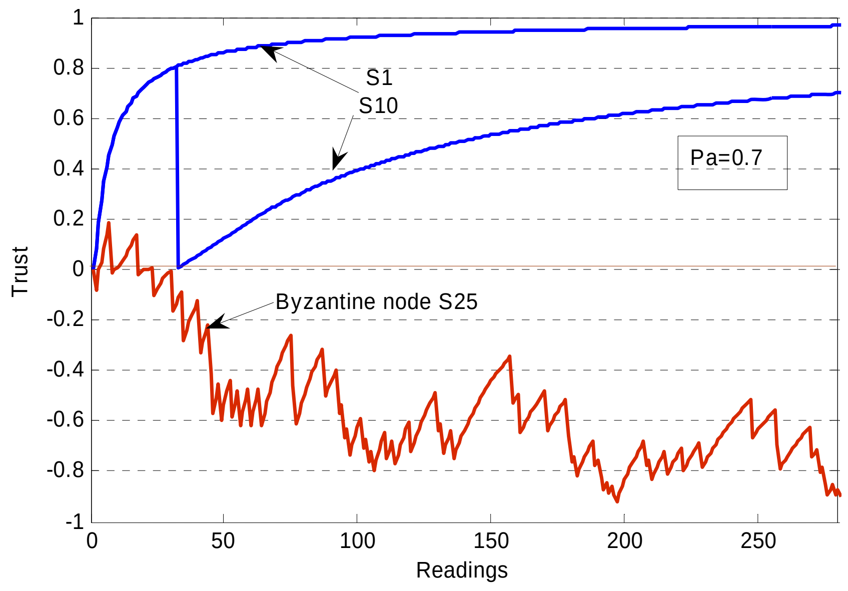


Figure 4-11 Sensor Node Trust (β =0.58)

The previous two diagrams do not consider the second threshold value, against which a sensor is assessed and, accordingly, can be excluded from the network. with inclusion of the second threshold, trust evaluation behaves differently. As Figure 4-12 shows, when a sensor's trust value drops below the first threshold value, cryptography is enforced on that sensor. If the drop of trust is due to an outside attack, then with cryptography on, the attack is mitigated and so the trust goes up again as in the case of S_{24}. However, if cryptography does not seem to solve the problem as the trust continues to decrease reaching the second threshold value, the sensor is removed.

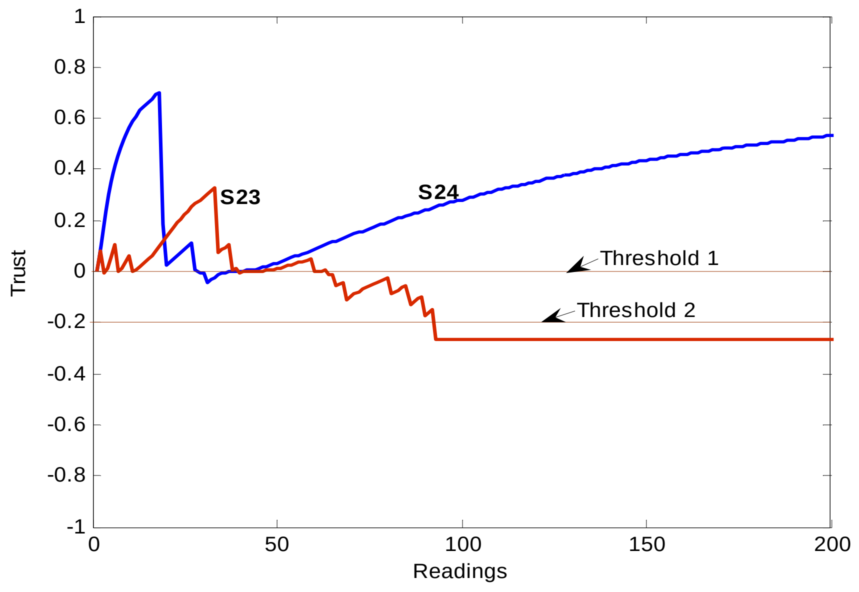


Figure 4-12 Trust After Enforcing Two Thresholds

However, it is also of interest to look at the outlier detection algorithm (GESD) performance. Based on Figure 4-13, a number of observations can be made. The first point is linked to the number of Byzantine nodes. As more of them contribute to the collection of information, the detection rate decreases, due to the masking problem in outlier detection. The problem happens when malicious data is not detected because of the existence of other outliers. As seen in Figure 4-13, when there is only one Byzantine node, the detection rate is at 100%. By increasing the number of Byzantine nodes to 7, the success rate drops to 86% (deviation of up to 10) and 80% (deviation of up to 5).

Second, looking at the figure from a different perspective, it can be concluded that the number of honest nodes have an influence on GESD. The detection rate improves with an increase of good nodes in the network. In the simulation, when the number of honest nodes is at 70% of the total number of nodes, the success rate is at 80%. With the increase of good nodes to 93%, the success rate reaches 100%.

Third, as Byzantine node measurements fall further away from the true mean value, the outlier detection rate becomes better. As shown in Figure 4-13, the successful detection rate shows an improvement of over 10% when the curve that simulates nodes reporting measurements up to 5 standard deviations from the mean is compared with the curve whose nodes report measurement further away from the mean (~10 standard deviations).

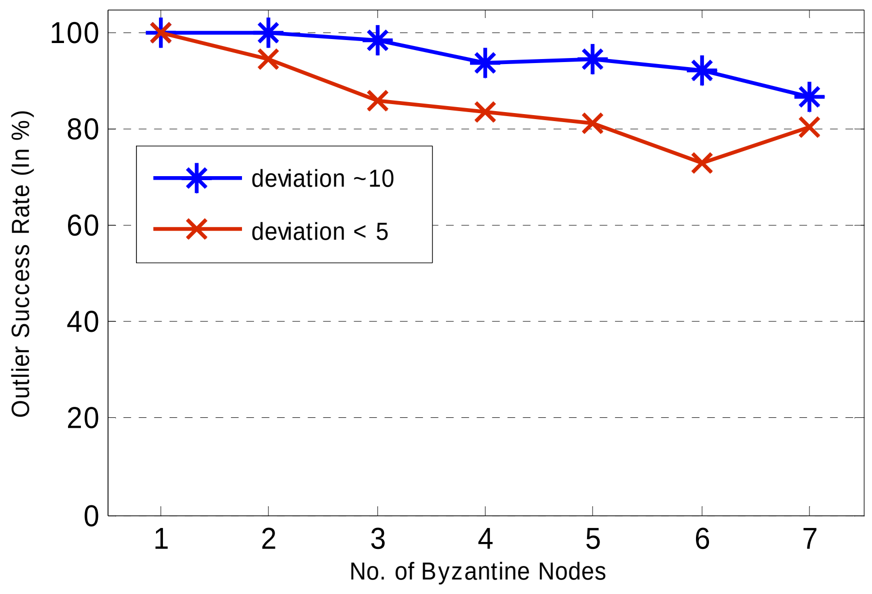


Figure 4-13 GESD Successful Detection Rate

The False Positive Rate can be calculated as the number of the nodes marked Byzantine mistakenly over the total number of good nodes in the system. From the simulation, results show that the false positive rate is zero. Figure 4-10 confirm that S_1-S_{18} are good nodes and S_{19}-S_{25} are Byzantine ones. However, as can be inferred from Figure 4-9 and Figure 4-10, although some of the sensor readings, such as S_4, S_{10}, and S_{15}'s, have been identified as outliers on certain occasions, none of them is marked Byzantine. In fact, the reason for some measurements possibly being reported as outliers can be related to noise and/or the sensor's position. As far as GESD is concerned, the false outlier detection rate is almost zero.

The Accuracy Improvement Rate can be calculated in terms of the achieved improvement to the standard deviation using the following equation.

$$\text{Accuracy Improvement Rate} = \frac{S_{bad} - S_{robust}}{S_{robust}} \times 100 \quad \%$$

where S_{bad} is the standard deviation with the existence of Byzantine nodes.

S_{robust} is the standard deviation after filtering out the Byzantine nodes.

Accuracy improvement is introduced mainly to prove the effectiveness of the GESD algorithm. As malicious data is excluded, the standard deviation becomes smaller, which improves the overall accuracy of the collected data. Figure 4-14 shows that up to 3 times the accuracy improvement can be gained when Byzantine nodes inject false data of up to 10 standard deviations.

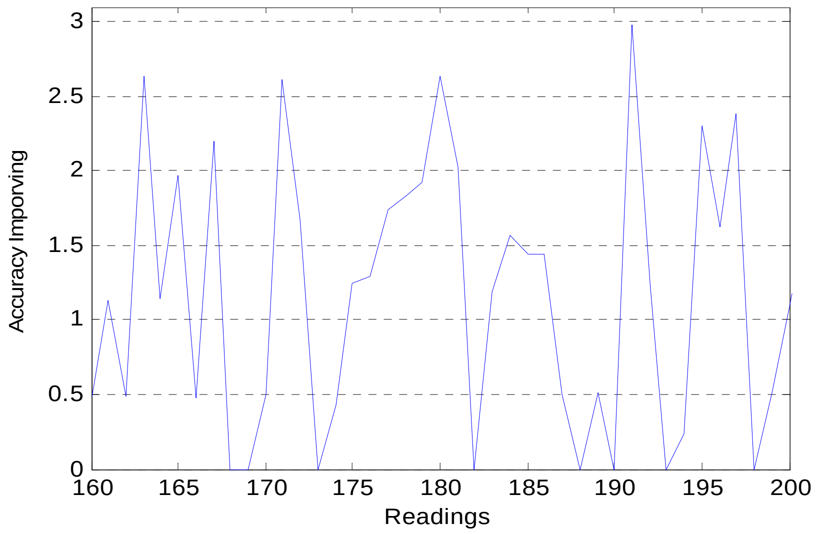


Figure 4-14 Accuracy Improvement (with 7 Byzantine Nodes)

However, the improvement rate is subject to the severity of malicious data. As Byzantine nodes report measurements that are further apart from other sensors consensus, excluding those malicious data can significantly improve the accuracy improvement rate. Furthermore, as the number of Byzantine nodes increases, the accuracy improvement rate increases. Figure 4-15 proves those two points. As can be noticed, when the deviation from consensus is around 10, the improvement average reaches 1.2 and it reaches 0.4 for 5 deviated data.

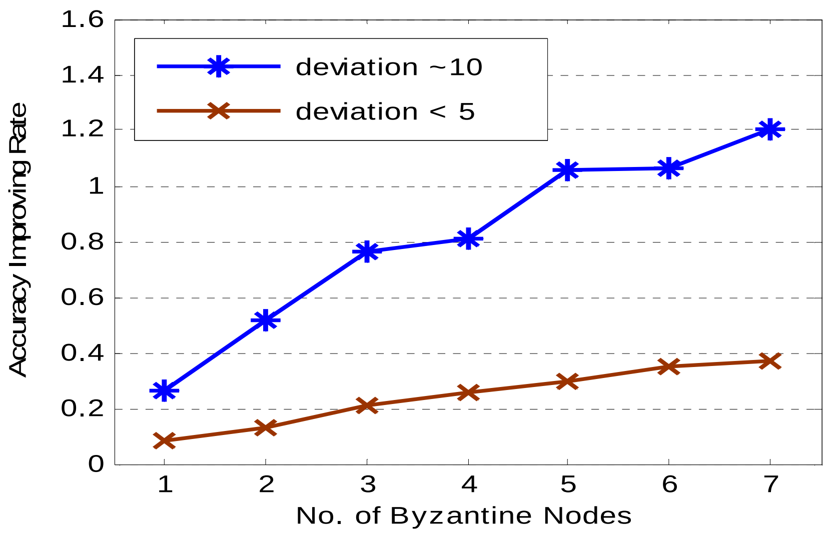


Figure 4-15 GESD Accuracy Improvement Rate

4.4.3 Energy Consumption

One contribution in favour of our security scheme is the conservation of energy it makes. As has been shown in Chapter 3, cryptography causes considerable extra consumption of energy, mainly due to packet overhead, which leads consequently to a shorter network lifetime. Our method reduces the impact of cryptographic tools by postponing their use until an attack is suspected. More precisely, cryptography is switched on to only those sensors whose trustworthiness value is low. Thus, this method saves 14% to 20% of power dissipation on every packet transmitted. The exact amount of energy saved depends on the security requirements, encryption and/or authentication, and the implemented cryptographic primitives, such as RC5, RC6, and DES.

For a better idea of how much power is expected to be saved, this section shows the amount of energy consumed in the case of SPINS [14] and TinySec [32] security algorithms (Chapter 3, Section 3.2.2). Both of them build their security blocks on top of the TinySec operating system, which constructs packets of size 36 bytes.

SPINS adds zero bytes for encryption but adds 6 bytes to every packet for authentication. Table 3 demonstrates the costs of computation and communication in terms of energy. Most of the overhead is related to the transmission of the extra bytes rather than computations.

Table 3 Energy Costs of Adding Security Protocols

Packet Component	Energy Consumption (%)
Data Transmission	71
Encryption Computation	<1
Encryption Transmission	<1
Freshness Computation	<1
Freshness Transmission	7
MAC Computation	2
MAC Transmission	20

Table 4 lists the TinySec security options and the corresponding energy consumption that is related to packet transmission. As can be seen, authentication and encryption require 5 bytes of overhead, leading to high energy consumption.

Table 4 Radio Energy Costs

Security Option	# of Overhead Bytes	Energy (mJ)	Increase (%)
No security	-	1.215	-
Authentication	1	1.247	2.6
Authentication and encryption	5	1.385	13.99

Regarding the energy consumed on CPU processing, every cryptographic primitive requires a different amount of time and a different number of CPU cycles for execution, resulting in different energy consumption values. Table 5 and Table 6 present the time, CPU cycles, and Energy consumed by ciphers and MAC algorithms.

Table 5 Costs of Ciphers on CPU

Algorithm	Time(ms)	CPU cycles	Energy(μJ)
SkipJack	2.16	15925.2	51.84
RC5	1.50	11059.2	36.00
DES	608.00	4,482,662.4	14,592.00

Table 6 Costs of MAC on CPU

Algorithm	Time(ms)	CPU cycles	Energy(μJ)
SkipJack	299	22,044.6	71.76
RC5	2.08	15,335.4	49.92
DES	1,208.00	8,906,342.4	28,992.00

In conclusion, Power efficiency is an important aspect, which directly influences network lifetime. By making the security choice and looking at the tables above, the security designer can estimate the amount of energy to spend. In our security scheme, enabling cryptography is directly related to the trust value a sensor node has. When the trust value drops lower than a defined threshold value, security is turned on immediately. Therefore, the following relation holds. By lowering the trust threshold value, sensors work without security for a longer time. As a result, the lifetime is extended, but security is relatively sacrificed. On the other hand, by raising the trust threshold value, sensor security is tightened, as sensors may start security earlier than in the previous case. This leads to a shorter lifetime but securer system.

CHAPTER 5 CONCLUSIONS AND FUTURE WORK

5.1 Summary and Contributions

Sensor networks promise viable solutions to many problems in a variety of fields. Sensing technology today is moving relatively fast from research contexts to industrial and social contexts, and with increased interest in implementing sensor networks, there comes a vital concern about data secrecy. Most applications, if not all, must be provided with security as an intrinsic. In addition, data aggregation techniques are highly desirable to prolong network lifetime and increase information accuracy.

The motivation behind this thesis is to relax the conflict that applying security on sensor networks tends to compromise other important issues. First, Cryptographic tools cause extra consumption of energy. Second, cryptographic functions assume that nodes are trustworthy as long as they use the assigned secret keys. Third, end-to-end security prevents intermediate nodes from modifying message contents. Consequently, applying security does not allow data aggregation techniques to take place, deprives sensor networks of a long lifetime, and does not solve the inside attack problem. In spite of all that, security and data aggregation must both be implemented because they are vital for the success of sensor networks. In this context, this thesis addresses security issues in wireless sensor network, with a strong focus on secure data aggregation. A novel mechanism is proposed to achieve data aggregation while maintaining security requirements and preserving energy, even in the presence of

Byzantine nodes (inside attacks). In the proposed technique, the aggregator, in addition to performing the regular aggregation function, identifies the outlying sensor nodes through an outlier detection algorithm called Generalized Extreme Studentized Deviate (GESD). Furthermore, it assigns information-theory-derived trustworthiness values to all sensors to keep track of their performance. Trust values are checked against two threshold values. When a sensor's trust is lower than the first threshold value, only then is cryptography enabled. When the value is lower than the second threshold, the sensor is assumed to be Byzantine and, therefore, it is excluded from the network.

In other words, the described mechanism has two novel contributions: 1) cryptography is switched on only when an attack is suspected, which preserves the energy that otherwise would be needed to run the cryptographic functions. The format of packets with and without cryptographic fields is also discussed, showing that 14% to 20% of overhead per packet can be saved. 2) data aggregation continues to work, even more efficiently, as malicious data and Byzantine nodes are suppressed. Simulation results show that the outlier detection procedure is able to successfully detect outliers with a high rate of 80% to 100%, depending on the number of Byzantine nodes, and with hardly any false alarms. Furthermore, nodes can be detected as fast as in 20 to 50 readings, depending on the configurable parameter, probably of attack P_A.

However, it should be noted that the proposed mechanism has some limitations. Firstly, the maximum number of Byzantine nodes that this mechanism can simultaneously handle must be less than half of the total number of nodes. Secondly, it assumes that aggregator nodes are trustworthy. Thus, for this mechanism to work efficiently, the aggregators must be provided with a higher level of security, such as tamper-resistant packaging, and be placed in secure locations. Thirdly, a master key is used in deriving other keys for all sensors to use. This keying technique introduces a shortcoming: if the master key is compromised, then the whole network can be compromised.

5.2 Future work

The work presented in this thesis can still have various extensions and directions for future work. One of the future works is to add the aggregator node to the list of non-trustees. In this thesis, the aggregator is regarded as a trusted node, which satisfies many sensor network applications. However, it is also of interest to determine the aggregator's honesty. For that, another mechanism should be added. Chapter 3 introduces some work done in this area, detecting a malicious aggregator. For example, Deng et al. [50] and Wu et al. [51] propose having watchdog-like nodes to monitor the aggregators. These techniques can be manipulated so that they integrate with my technique. Moreover, an interactive proof technique in which the home server ensures that the aggregator is not malicious is possible. That is, the home server investigates previous readings and assigns trust values to the aggregator based on them.

A second possible approach for extension is to implement multi-tiered security architecture. The proposed scheme assumes that cryptography is either on or off. With multi-tiered security design, different levels of security can be maintained. Every security level can be triggered in accordance with the trust assessment. For example, security can be set to the minimum when the trust value is high, but as the trust value slides down, a certain level of security is enforced. Moreover, when the number of Byzantine nodes increases, all sensors are forced to increase their security level. By offering a range of security levels, the limited sensor resources are better utilized according to the required protection level and severity of the threat. This approach requires studying the cryptographic primitives to select the ones that satisfy the goal. A preliminary survey shows that RC5 and RC6 are possible candidates as they are fully parameterized.

To summarize, security protocols and data aggregation techniques seem to introduce conflicts; however, integrating them both is essential for the success of a sensor network. The results of this thesis provide a good starting point for a deeper study of secure data aggregation protocols.

References

[1] On world at http://www.onworld.com/html/wirelesssensorsrprts.htm

[2] C. Karlof and D. Wagner. Secure Routing in Sensor Networks: Attacks and Countermeasures. In Proc. of First IEEE International Workshop on Sensor Network Protocols and Applications, May 2003.

[3] H. Karl and A. Willig. Protocols and Architectures for Wireless Sensor Networks. Wiley, 2005. ISBN:0470095105.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci. Wireless Sensor Networks: a Survey. Computer Networks (Amsterdam, Netherlands: 1999)

[5] S. Tilak, N. Abu-Ghazaleh, and W. Heinzelman. A Taxonomy of Wireless Microsensor Network Models. ACM Mobile Computing and Communications Review 2002.

[6] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. Elsevier Ad Hoc Network, Vol. 3/3:pages 325-349, 2005

[7] J. N. Al-Karak and A. E. Kamal. Routing techniques in wireless sensor networks: a survey. IEEE Wireless Communications. Dec. 2004.

[8] I. F. Akyildiz, Y. Sankarasubramaniam, W. Su, and E. Cayirci. Wireless Sensor Networks: a survey. Computer Networks: The International Journal of Computer and Telecommunications Networking, Volume 38 Issue 4, March 2002.

[9] E. e. a. Sohrabi, “Protocols for self-organization of a wireless sensor network,” pp. 16–27, October 2000.

[10] A. Woo and D. Culler, “A transmission control scheme for media access in sensor networks,” July 2000.

[11] W. Heinzelman, J. Kulik, and H. Balakrishnan, “Adaptive protocols for information dissemination in wireless sensor networks,” pp. 174–185, 1999.

[12] C. Intanagowiwat, R. Govindan, and D. Estrin, “Directed diffusion: A scalable and robust communication paradigm for sensor networks,” pp. 56–67, 2000.

[13] C. Shen, C. Srisatjapornphat, and C. Jaikaeo, “Sensor information networking architecture and applications,” IEEE Pers. Communication, pp. 52–59, Aug. 2001.

[14] A. Perrig, R. Szewczyk, D. Tygar, V. Wen, and D. Culler, "SPINS: Security protocols for sensor networks", Wireless Networks Journal (WINE), September 2002.

[15] S. Avancha, J. Undercoffer, A. Joshi, J. Pinkston. Security for wireless sensor networks. ISBN: 1-4020-7883-8. pages 253-275. 2004.

[16] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem", ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982, pp 382-410.

[17] Radia Perlmen, Routing with Byzantine Robustness, Sun Microsystems, Sep2005

[18] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H Rubens, "Mitigating Byzantine Attacks in Ad Hoc Wireless Networks," Department of Computer Science, Johns Hopkins University, Tech. Rep. Version 1, March 2004.

[19] M. Pease, R. Shostak, L. Lamport, "Reaching Agreement in the Presence of Faults", JACM 27, 2, 228-234, 1980.

[20] M. Fischer and N. Lynch, "A Lower Bound for the Time to Assure Interactive Consistency". Information Processing Letters 14, 4, 183-186, 1982.

[21] M. Fischer, N. Lynch, and M. Paterson, "Impossibility of Distributed Consensus with One Faulty Process", JACM, 32, 2, 374-382, 1985.

[22] G. Guimaraes, E. Souto, D. Sadok, and J. Kelner, Evaluation of Security Mechanisms in Wireless Sensor Networks, Proceedings of the 2005 System Communications.

[23] M. Yu, S. Kulkarni, and P. Lau, "A New Secure Routing Protocol To Defend Byzantine Attacks For Ad Hoc Networks", IEEE Int. Conf. on Networks (ICON'05), vol. 2, pp. 1126-1131, Nov. 16-18, Kuala Lumpur, Malaysia.

[24] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, "An on-demand secure routing protocol resilient to byzantine failures," in ACM Workshop on Wireless Security (WiSe) 2002, 2002.

[25] G. Gaubatz, J.-P. Kaps, and B. Sunar, "Public key cryptography in sensor networks revisited", 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004),2004.

[26] H. Cam, S. Ozdemir, D. Muthuavinashiappan, and P. Nair, "Energy-efficient security protocol for wireless sensor networks", in Proc. of IEEE VTC Fall 2003 Conference, October 4-9, Orlando, 2003.

[27] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A link layer security architecture for wireless sensor networks", Proc. of the Second ACM Conference on Embedded Networked Sensor Systems, November 3-5, Baltimore, 2004.

[28] A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography, volume 6 of Discrete Mathematics and Its Applications. CRC Press,

Greenwich, Connecticut, 1996.

[29] Y. Law, J. Doumen, and P. Hartel. Survey and benchmark of block ciphers for wireless sensor networks. Tech. Rep. TRCTIT- 04-07, Centre for Telematics and Information Technology, University of Twente, The Netherlands, 2004.

[30] N. Gura, A. Patel, A. Wander, H. Eberle, and S. Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In 2004 workshop on Cryptographic Hardware and Embedded Systems, Aug. 2004.

[31] A. Perrig, R. Szewczyk, D. Tygar, V. Wen, and D. Culler, "SPINS: Security protocols for sensor networks", Wireless Networks Journal (WINE), September 2002.

[32] C. Karlof, N. Sastry, and D. Wagner. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. Proceedings of the 2nd international conference on Embedded networked sensor systems, pp 162-175, November 2004.

[33] A. D. Wood and J. A. Stankovic, Denial of service in Sensor networks, computerm Vol. 35, no. 10, pp. 54-62, 2002.

[34] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. Computer, 35(10):54–62, 2002.

[35] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in Mobile Computing and Networking, 2000, pp. 243–254.

[36] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Wormhole detection in wireless ad hoc networks," Department of Computer Science, Rice University, Tech. Rep. TR01-384, June 2002.

[37] J. R. Douceur, The Sybil Attack, in 1st International Workshop on Peer-to-Peer Systems (IPTPS 02), March 2002.

[38] S. Avancha, J. Undercoffer, A. Joshi, and J. Pinkston, "Secure sensor networks for perimeter protection," Computer Networks Wireless Sensor Networks, vol. 43, no. 4, pp. 421–435, 2003.

[39] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva, "Directed Diffusion for Wireless Sensor Networking", IEEE/ACM Transactions on Networking, vol. 11, no. 1, February 2003.

[40] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks", Proc. of 5th ACM/IEEE Mobicom Conference, 1999.

[41] M. Qin and R. Zimmermann, "An energy-efficient voting-based clustering algorithm for sensor networks," in ACIS Int. Workshop Self-Assembling Wireless Networks(SAWN 2005), May 2005.

[42] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor network,” IEEE Trans. on Wireless Communications, vol. 1, pp. 660–670, Oct 2002.

[43] O. Younis and S. Fahmy, “Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach,” in Proceedings of IEEE INFOCOM, March 2004.

[44] M. Chatterjee, S.K. Das and D. Turgut, “WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks”, Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks), Vol. 5, No. 2, pp. 193-204, April 2002.

[45] S. Basagni. Distributed Clustering for Ad Hoc Networks. In Proc. Of I-SPAN, pages 310–315, 1999.

[46] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong, “TAG: A tiny aggregation service for ad-hoc sensor networks,” in OSDI, 2002.

[47] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. In SenSys ’04: Proceedings of the 2nd international conference on Embedded networked sensor systems, pages 239–249. ACM Press, 2004.

[48] L. Hu and D. Evans, "Secure aggregation for wireless networks", Proc. of Workshop on Security and Assurance in Ad hoc Networks, Jan 28, Orlando, FL, 2003.

[49] R. C. Merkle, "Protocols for public key cryptosystems", Proc. of the IEEE Symposium on Research in Security and Privacy, April 1980, pp. 122- 134.

[50] W. Du and J. Deng and Y. S. Han and P. K. Varshney, "A Witness-Based Approach for Data Fusion Assurance in Wireless Sensor Networks", in Proc. of IEEE Global Telecommunications Conference (GLOBECOM '03), pp. 1435-9,2003.

[51] K. Wu, D. Dreef, B. Sun, and Y. Xiao, "Secure Data Aggregation without Persistent Cryptographic Operations in Wireless Sensor Networks", Proc. of 25th IEEE International Pe@ormance, Computing, and Communications Conference, (IPCCC) 2006. , pp. 635-640,2006.

[52] Donggang Liu and Peng Ning, “Establishing pairwise keys in distributed sensor networks,” in Proceedings of ACM CCS, October 2003.

[53] S. Zhu, S. Setia, and S. Jajodia, “LEAP: Efficient security mechanisms for large-scale distributed sensor networks,” in Proceedings of ACM CCS, October 2003.

[54] W. Zhang, H. Song, S. Zhu, and G. Cao, “Least Privilege and Privilege Deprivation: Towards Tolerating Mobile Sink Compromises in Wireless Sensor Networks,” ACM MobiHoc, May 2005.

[55] D. M. Hawkins, Identification of Outliers, New York: Chapman and Hall, 1980.

[56] B. Rosner, “Percentage points for generalized esd many-outlier procedure,” Technometrics, 1983.

[57] J. E. Seem, G. Wi, Method of Intelligent data Analysis to Detect Abnormal Use of Utilities in Buildings, US patents, Patent no. US6816811 B2, Nov 2004.

[58] H. Song, S. Zhu, G. Cao, Attack-resilient time synchronization for wireless sensor networks, in IEEE MASS, 2005, pp765-772.

[59] Y. Sun, W. Yu, and K. Riu. Trust Modelling and Evaluation for Ad Hoc Networks, GLOBECOM IEEE, Dec, 2005.

[60] A. Becher, Z. Benenson, and M. Dornseif. Tampering with motes: Real-world physical attacks on wireless sensor networks. In 3rd International Conference on Security in Pervasive Computing (SPC), April 2006.

[61] D. Wagner, ResilientAaggregation in Sensor Networks, Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, 2004

[62] Z. Liang and W. Shi. PET: A PErsonalized Trust model with reputation and risk evaluation for P2P resource sharing. Proceedings of the HICSS-38, Jan. 2005.

[63] S. Slijepsevic, M. Potkonjak, V. Tsiatsis, S. Zimbeck, and M. Srivastava, “On communication security in wireless ad-hoc sensor networks,” in Proc. 11th IEEE Int. Workshops Enabling Technol.: Infrastructure for Collaborative Enterprises, Jun. 2002,

[64] B. Przydatek, D. Song, and A. Perrig, "SIA: Secure information aggregation in sensor networks", Proc. of SenSys'03, Nov 5-7, Los Angeles, CA, 2003.

[65] J. Deng, R. Han, and S. Mishra. Countermeasuers against traffic analysis in wireless sensor networks. Technical Report CU-CS-987-04, University of Colorado at Boulder, 2004.