Secure Routing and Data Aggregation for Infrastructureless Wireless Networks without Persistent Cryptographic Operations

by

Dennis Sebastian Dreef
B.Sc, University of Victoria, 2004

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Secure Routing and Data Aggregation for Infrastructureless Wireless Networks without Persistent Cryptographic Operations

by

Dennis Sebastian Dreef

Supervisory Committee

Dr. Kui Wu, Supervisor (Department of Computer Science)

Dr. Jianping Pan, Department Member (Department of Computer Science)

Dr. Valerie King, Outside Member (Department of Computer Science)

Dr. Issa Traore, External Examiner (Department of Electrical and Computer Engineering)

Supervisory Committee:

Dr. Kui Wu, Supervisor (Department of Computer Science)

Dr. Jianping Pan, Department Member (Department of Computer Science)

Dr. Valerie King, Outside Member (Department of Computer Science)

Dr. Issa Traore, External Examiner (Department of Electrical and Computer Engineering)

ABSTRACT

Nodes in infrastructureless wireless networks usually have only limited energy and pose new security challenges, since traditional cryptographic operations have a high energy cost. In this thesis, new security solutions are presented to avoid using costly cryptographic operations. Two secure problems are investigated: the first is secure routing; the other is secure data aggregation. For the first problem, a randomized algorithm is proposed to defend against malicious attackers wishing to disrupt routing in wireless ad-hoc networks. For the second problem, a solution is proposed to leverage the broadcast nature of wireless medium and use a special aggregation topology, namely a clique tree, for data integrity in wireless sensor networks. With analysis and performance evaluation, both solutions are demonstrably lightweight with acceptable security features.

Table of Contents

List of Tables

List of Figures

Acknowledgement

I would like to acknowledge the support and help of many people. My supervisor, Dr. Kui Wu, has helped me immensly with this thesis and without his help this would not have happened. My committee for their careful review and probing questions. I would also like to thank Dianne, for putting up with me, keeping me sane, and supporting me as I persued my Masters. A special thanks to my parents and familly, who supported me in many ways. The other grad students that I shared my time with in TEF. The District 5 crew for giving me an outlet Monday nights. And there are many others who have made my time at UVic something to remember and helped me accomplish all that I have, so thank you for everything.

Chapter 1

Introduction

1.1 Motivation

With the ever-increasing prevalence of wireless enabled devices and the lowering cost of Micro-Electro-Mechanical Systems (MEMS), new areas of research have emerged. Mobile Ad-Hoc Networks (MANETs) form one such area, while Wireless Sensor Networks (WSNs) another. MANETs typically consist of mobile nodes that communicate via a wireless medium and do not depend on existing infrastructure such as routers, switches, bridges or access points. A MANET is formed based on a nodes location and the need to share data among the other nodes. A WSN is formed with MEMS based sensor devices that are placed or scattered in an area of interest to gather data unobtrusively. In both types of networks, a fundamental problem is how to deliver data from one node to another while maintaining data integrity, authenticity, and confidentiality.

Figure 1.1 shows an example of a MANET, with several nodes linked together wirelessly. It is important to note that MANETs can be heterogeneous since many different wireless devices, such as PDAs, laptops, and Bluetooth-enabled cellular phones, can be used. Figure 1.1 also illustrates an important phenomenon: not all nodes are within the radio range of each other, thus there is a need for multi-hop routing. Multi-hop routing is the process of finding a route from one node to another node that is potentially faraway, where intermediate nodes are required to forward the data between the two communicating nodes. The reliance on intermediate nodes introduces several security concerns related to the in-

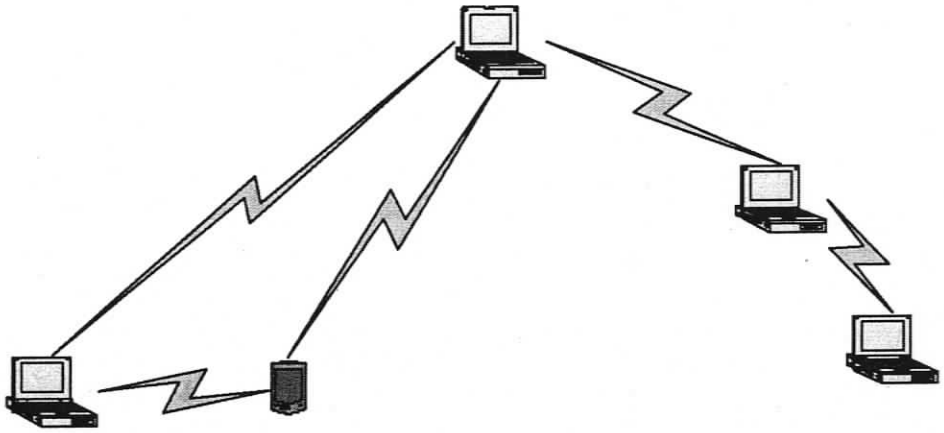


Figure 1.1. *An example of an Ad-Hoc Network*

tegrity, authenticity and confidentiality of data, as the intermediate nodes may eavesdrop or alter the messages, or worse yet they may just pretend to be someone else.

Security in wireless networks is expensive, no matter whether it is solved via encryption, handshaking or other means. Since most nodes in MANETs are mobile and have limited battery life, we should expend energy in a conservative manner. Extraneous transmissions and massive computation deplete energy quickly and thus reduce the network lifetime and performance. In this sense, cryptography is not a silver bullet; it is an expensive means to an end.

The important services to the reliability and usability of multi-hop wireless networks include:

- Routing: Routing is a major component of MANETs, it is also an insecure mechanism as the majority of the protocols assume that neighbors are honest in their replies. A dishonest neighbor could wreak havoc on a network, illegitimately positioning itself in the route among other things.
- Aggregation: In WSNs, in-network data aggregation is a commonly used approach to reducing message transmission. The concern here is if a node is compromised, the node can change data and violate the integrity of information. An aggregation tree

should be constructed in such a way that it can provide the data integrity at a low cost.

1.2 Secure Routing

There are two predominant strategies for routing: proactive routing and re-active routing. The first type of routing protocol collects the routing information for any possible destination within the network, regardless of whether there exists data traffic to the destination. This means that there is a known path from any given node to another. The problem with proactive protocols is that in a dynamic MANET, routes may be changed frequently, resulting in heavy control overhead. So for MANETs there is a very desirable alternative, reactive routing or on-demand routing, that finds a route to a host only when the route is needed. Compared to proactive routing, reactive routing can dramatically reduce routing overhead because it does not need to search for and maintain the routes on which there is no data traffic. This property is very appealing in the resource-limited environment.

Compared to wireline networks, wireless networks are more easily attacked due to the unshielded transmission media. In MANETs, each node relies on other nodes to relay its messages to the ultimate destination. A compromise of an intermediate node may result in severe security breaches. For instance, the compromised node may alter messages, disrupt message flows, or form a black hole by using fake small hop-count values. The compromise of routing protocols will finally paralyze the whole network.

In this thesis, we will focus on adding security mechanisms to existing on-demand routing protocols [1, 2]. Due to stringent constraints on energy supply to wireless nodes, we focus on lightweight secure solutions, i.e., solutions with small overhead on calculation and communication, and the tradeoff between security and the system cost. This feature makes our work significantly different from existing secure solutions for the Internet where complex and intensive cryptographic calculations are generally used.

1.3 Secure Aggregation

We look at the security of data aggregation in WSNs, and use topology constraints on the aggregation tree (or route) that can detect cheating and ensure data integrity.

Extremely small sensors have been used broadly for various applications such as habitat monitoring [3], battlefield surveillance, and forest fire monitoring. A large number of tiny sensors collect measurement data and rely on multi-hop short-range radio communication to send data to the processing center, which is also called the base station or sink node. The major motivation to use tiny sensors is to save energy consumption and reduce system cost, as recent tiny sensors can work without recharging for several months. The long life of sensors, however, depends on effective energy saving strategies such as sensor scheduling and information processing within the network to reduce the data traffic to the base station. One important form of in-network processing is data aggregation [4].

Depending on different application contexts, data aggregation could be simply compression of the data or the calculation of some statistical values, such as mean, median, max, and min. While data aggregation can effectively reduce the amount of data transmitted to the base station, raw data items are invisible to the base station and thus their authenticity and integrity are hard to guarantee. As such, data aggregation provides a good opportunity for attackers to inject bogus information or forge aggregated information without being detected easily. For instance, if a sensor close to the base station is compromised, it may send fake aggregation values to the base station and mislead the base station into trusting the falsified information. It may have disastrous impact if end users respond according to the faulty information.

To simplify the problem, we assume that the confidentiality of data is not a major concern in WSNs and beyond the focus of this thesis. Instead, the integrity and the authenticity of data must be guaranteed. A secure sensor network must be able to effectively defend against an adversary who can compromise sensors and inject bogus information into the network. Such an attacker could mislead the network operator into an erroneous response

or cause the monitoring system to fail to detect an intrusion.

Providing security to sensor networks is extremely challenging due to the stringent constraints on energy supply and computing capability. Traditional security techniques such as public-key cryptography, although very strong, require intensive calculation and thus are not suitable for sensor networks. This thesis focuses on lightweight security mechanisms and tradeoffs between security and the cost of bandwidth and energy.

1.4 Contributions

The goal of this work was to limit the persistent or on-going and continuous use of cryptography and with the major contributions listed below we show this is possible. The major contributions of this thesis include:

- The introduction of Intrusion Detection results into the routing process. The use of information as a parameter in the decision making process along with a randomization element showed significant increase in resiliency to attacks in AODV (Ad-hoc On-demand Distance Vector) [1] routed MANETs.
- A new tree construction mechanism in WSNs. This tree building algorithm is based on the desire to introduce in-network cheating detection as a deterrent; while also saving energy costs, as expensive cryptography is replaced by watchdogs.
- A thorough evaluation of the performance of the above mechanisms, regarding their overhead and security.

1.5 Outline of the Thesis

In the rest of the thesis we will present and analyze a set of protocols that enhance security with small overhead in their respective domains. In Chapter 2 we introduce the background material and related work. In Chapter 3 we describe our protocol enhancement to AODV [1], including performance analysis, simulation results, and implementation details.

In Chapter 4 we present our second protocol and provide a full description of how it works, and the theory behind it. We also present the simulation results and implementation details. Finally we conclude the thesis and introduce future work in Chapter 5.

Chapter 2

Background

2.1 Secure Routing in MANETs

2.1.1 Routing in MANETs

Routing in MANETs is difficult since mobility causes frequent network topology changes and requires more robust and flexible mechanisms to search for and maintain routes. When the network nodes move, the established paths may break and the routing protocols must dynamically search for other feasible routes. With a changing topology, even maintaining connectivity is very difficult. In addition, keeping the routes loop free is more difficult when the hosts move. Besides handling the topology changes, routing protocols in MANETs must deal with other constraints, such as low bandwidth, limited energy consumption, and high error rates, all of which may be inherent in the wireless environment. Furthermore, the possibility of asymmetric links, caused by different power levels among mobile hosts and other factors such as terrain conditions, make routing protocols more complicated.

Many protocols have been proposed to solve the above problems. These protocols can be divided into two main categories: *proactive* routing and *reactive* routing. Proactive methods maintain routes to all nodes, including nodes to which no packets are sent. Such methods react to topology changes, even if no traffic is affected by the changes. They are also called table-driven methods. Reactive methods are based on-demand for data transmission. Routes between hosts are determined only when they are explicitly needed to

forward packets. Reactive methods are also called on-demand methods.

Compared to proactive routing protocols, reactive routing protocols can significantly reduce routing overhead when the traffic volume is light and the topology changes less dramatically, since they do not need to update route information periodically and do not need to find and maintain routes on which there is no traffic. This property is very appealing in the resource-limited environment. In this thesis, we use a popular reactive routing protocol, AODV (Ad-hoc On-demand Distance Vector routing [1]), to illustrate our security solution.

2.1.2 AODV

AODV (Ad-hoc On-demand Distance Vector routing [1]) works by discovering a route by broadcasting a route request message (RREQ) with a unique ID to its neighbors. The neighbors create an entry in their routing table for the first RREQ with the given ID. All proceeding RREQ packets with the same ID are considered duplicates and are dropped. Each neighbor subsequently broadcasts the RREQ and this procedure continues until the RREQ reaches the destination or a node with a valid path to the destination, where a valid route is one whose sequence number is greater than or equal to the sequence number included in the RREQ.

The sequence numbers are maintained by each node. The purpose of using sequence numbers is to ensure loop freedom and maintain up-to-date information in their respective routing tables. All route information is kept in the route table. The routing table for AODV is shown in Table 2.1.

Once the destination node or a node with a valid route to the destination has received the RREQ, it will create a route reply message (RREP) intended for the source. The RREP will be uni-cast along the reverse path maintained by a precursor list. A valid path from the source to the destination will be established when the RREP message arrives at the source node.

Route maintenance is done with Route Error (RERR) messages. Upon detection of a link break in an active route, the node sends RERR messages to the nodes in its precursor

Table 2.1. *AODV Routing Table*

Field	Description
Destination IP address	The IP address of the destination for which a route is desired
Destination sequence number	The latest sequence number received in the past by the node
Valid destination sequence number flag	Indicates the validity of the sequence number
Route state	Consists of "valid", "invalid", "repairable" and "being repaired" states used to indicate the status of the route
Hop count	Number of hops needed to reach the destination
Next hop IP	The proceeding hop IP in the forward path
Lifetime	Expiration or deletion time of the route
Precursor list	List of nodes for whom this node is a forwarding node

list. Once a node receives an RERR message from its neighbor, it consequently forwards the RERR message to its precursor list.

2.1.3 Security Threats to MANET Routing

Ad-hoc routing protocols are very susceptible to the man-in-the-middle attack. That is in contrast to wired networks. The fact that communication occurs via a broadcast medium allows for easy eavesdropping. Add into the mix that neighbors are relied on for honesty and man-in-the-middle attacks can be performed fairly easily. A commonly described attack on routing protocols is the so-called Black Hole [5] attack, also known as the Worm Hole [6] attack. This attack can be used as either a denial of service or method of positioning oneself as the man in the middle. A node would just advertise or respond to route requests with a very short route to the desired node, thus giving it a place in the path. Once a node has cemented itself in as part of the route, it can listen in, change the content, or even drop all packets.

In the following, we introduce a specific type of man-in-the-middle attack, rushing attacks, and the existing solutions to rushing attacks. This thesis will present a new solution to defend against this type of attacks in Chapter 3.

2.1.4 Rushing Attacks to MANET routing

The Rushing Attack [7] is an attack against on-demand routing protocols. In on-demand routing, a node requiring a route to a destination floods the network with RREQ packets in an attempt to find a path to the destination. To limit the overhead of this flood, each node typically forwards only one RREQ originating from any route discovery. In particular, existing on-demand routing protocols, such as AODV, only forward the RREQ that arrives first from each route discovery.

In the rushing attack, the attacker exploits the aforementioned property of forwarding RREQs in a predictable manner. It increases its chances of being included in the path by

using various methods of advantageous packet forwarding. What follows are descriptions of three variations of the rushing attack as presented in [7].

MAC Layer Rushing Attack: Medium Access Control (MAC) protocols generally impose delays between the time when the packet is handed to the network interface for transmission and when the packet is actually transmitted. Because collision detection for broadcast packets is difficult, on-demand protocols specify a random delay between receiving a request and forwarding it to avoid collision. An attacker ignoring delays at either the MAC or routing layers will generally be preferred over similarly located non-attacking nodes, increasing the chances of establishing a path via the attacker.

Spurious Packet Flooding: Advantage in forwarding speed can also be achieved by keeping the buffers of neighboring nodes full. If each node processes the packets it receives in order and if an inadequate request authentication mechanism is used, a node can be kept busy processing spurious packets, thus deteriorating its ability to forward legitimate requests. Protocols employing public key techniques are particularly subject to these attacks, since they require considerable computation to verify each received request.

Wormhole and High Transmission Power: Faster transit of RREQs can be achieved by transmitting packets at a higher wireless transmission power level and reducing the number of nodes that must forward that RREQ to arrive at the target. A more powerful variant of this attack occurs when a tunnel is formed by two attackers where the first attacker simply forwards all control packets to the other attacker which subsequently broadcasts them. If the tunnel provides significantly faster transit than legitimate forwarders, nodes near one end of the tunnel generally will be unable to discover routes to the other end of the tunnel that do not include the tunnel. In general, a wired tunnel will provide faster transit than native, multi-hop wireless forwarding.

2.1.5 Existing Solutions to Rushing Attacks

The proposed solution to rushing attacks in [7] is a combination of two main mechanisms: random message forwarding and secure route discovery.

Random Message Forwarding: In traditional RREQ forwarding, the receiving node forwards the first RREQ and suppresses all subsequent RREQs. In the method proposed in [7], the node first collects a number of RREQs, and then selects an RREQ at random to forward. This method requires two parameters, the number of RREQs to be collected, n, and timeout value, t. Each node collects the maximum possible number of requests until either the maximum is collected or the timeout period is reached.

Secure Route Discovery: This mechanism [7] is based on the fact that legitimate nodes forward only one RREQ in any route discovery. The solution has three main parts:

- Secure Neighbor Detection protocol allows two nodes to detect each other as neighbors only if they can communicate and are within maximum transmission range of each other. This mechanism prevents an attacker from claiming to be a neighbor of another node when it is outside its maximum transmission range. This consists of a three-round mutual authentication protocol that uses delay timing to ensure that the other party is within communication range. The initiator sends a Neighbor Solicitation packet and then receives a reply. The initiator then sends a Neighbor Verification. The delay between the Neighbor Solicitation and Neighbor Reply provides an upper bound on the distance of the neighbor.
- Secure Route Delegation is used to verify that all the secure Neighbor Detection steps were performed between any adjacent pair of nodes. This can be described with the following example:

  Consider two neighbors A and B. A has received an RREQ from node S destined for node R with sequence number, i. Node A performs the Secure Neighbor Detection protocol and verifies that B is its neighbor, then it delegates the RREQ to B by signing the RREQ and forwarding it to B. B verifies the signature and the process continues.
- Buffered requests need to be duplicate-suppression-unique; that is, if any two route requests contain node A, the route prefix leading up to (and including) A must be the same.

Though these solutions do prevent the rushing attack from being as effective as it could potentially be, it does also require heavy usage of precious bandwidth and processing power. From the three-way handshake required for secure neighbor detection to the message signing used in secure route delegation, this solution requires a lot of resources. Furthermore, it waits for an independently chosen timeout period before randomly forwarding one of the received RREQs, causing significant delay during route discovery.

2.1.6 Other Security Solutions

For ease of reference, we introduce several other approaches to securing MANETs. These solutions are significantly different from the proposals in this thesis in that they rely on strong and complex cryptographic operations to accomplish security goals.

Ariadne

Ariadne [8] is a secured routing protocol, that provides security with the use of symmetric cryptography. It claims to be resistant to node compromise. It provides three mechanisms to secure the routing protocol. The first is to share secrets between each pair of nodes. This mechanism may become very expensive in terms of space. The second is to share secrets between communicating nodes and use TESLA [9] for broadcast authentication between forwarding nodes. The third and least appealing one is to use digital signatures.

From the results in [8], Ariadne requires heavy calculation of hash chain and MAC (Message Authentication Code). In addition, this protocol requires some pre-existing mechanism for key distribution, using a Key Distribution Center that in some way circumvents the circular relationship between the key setup and the routing protocol. Furthermore when TESLA is used, there is a requirement for loose time synchronization. Although TESLA provides the ability to handle a difference in time of δ, a large δ value will potentially delay the construction of on-demand routes to the point where it is moot.

SEAD

SEAD [10] is a proactive, secure distance vector based routing protocol. We discuss

this work as it shows how a proactive system might be secured. The authors favor a one-way hash functions, and do not use asymmetric cryptography. The DSDV (Destination-Sequenced Distance Vector) routing protocol serves as the base for this extension. SEAD adds destination sequence numbers to avoid replay attacks, it also avoids the use of update delay as they would allow attackers to gain an advantage by not adhering to the delay. Each node uses a one-way hash chain and uses one element in each routing update it sends.

2.2 IDS

MANETs do not have concentration points such as gateways, switches and routers where the traffic can be easily monitored. Additionally the nodes typically have a limited local knowledge of the network topology unless OLSR [11] which is a proactive routing protocol is in use. These factors along with different communication patterns between the wired and wireless worlds make intrusion detection more challenging and eliminate the chance of using a mature wired IDS in the MANET environment.

Intrusion detection mechanisms can be classified into misuse detection and anomaly detection. Misuse detection systems use patterns of well-known attacks or weak spots of the system to match and detect known intrusions. While anomaly detection systems flag observed activities that differ significantly from normal usage profiles.

Zhang et al. [12] have proposed an anomaly-based model for intrusion detection in MANETs. The architecture involves an IDS agent for each node. These agents are responsible for:

- Data collection: This module collects various streams of data, system and user activities as well as communication activities within the transmission range, from disparate sources.
- Local detection: This module analyzes gathered data traces provided by the data collection module to detect anomalies through statistical anomaly detection techniques.
- Cooperative detection: A node that detects an anomaly with inconclusive evidence

can initiate cooperative detection by propagating information among its neighbors.

- Intrusion response: This is dependent on the intrusion type. Some options include re-initializing routes and reorganizing the network to preclude attackers.

Normal behavior patterns are gathered during a training process. During this process the normal profiles of data of interest are recorded, and the deviations from these profiles with some normal and abnormal activities are included. The nodes will continually adapt what they deem normal behavior based on the environmental factors and topology of the network from their point of view.

To detect abnormal updates to routing tables, the nodes monitor the percentage change of routes (PCR) and percentage change in sum of hop counts (PCH). The values in these fields will be compared to the expected value (gained through training and learning), and if it is within the threshold, behavior is assumed to be normal. If the value is not within the appropriate range, the system will generate a warning with a probability indicating the confidence in the attack.

The framework adopts the premise that normal activities have observable traits and that when detecting anomalies, the values of those traits stray from the norm hence indicating an attack. The problem of continuously classifying the behavior of the network is solved by using RIPPER [13] and SVM light [14]. RIPPER provides a decision tree and is used for rule induction. SVM light mines the data for patterns and classifies the results.

This framework can be extended to detect abnormalities at most layers of the protocol stack, ranging from the medium access control layer to the application layer. Given that there are vulnerabilities at all layers and each has its own intrusion detection module, these can be integrated to coordinated detection and response.

Nevertheless, due to mobility and the lack of centralized control point, MANET intrusion detection usually has a high false positive rate (a normal behavior is flagged as anomalous) that creates a conundrum for users. It is very hard for end users to justify an intrusion response based on unreliable information.

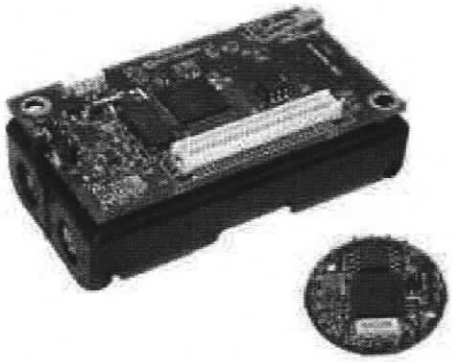


Figure 2.1. *Xbow MICA2 Mote and a MICA2DOT Mote(small round)*

2.3 Secure Data Aggregation in WSNs

2.3.1 The Limitations of Sensor Devices

Wireless Sensor Networks (WSN), are generally limited by battery lifetime and CPU power. Figure 2.1 depicts an example sensor device, Mica2 Mote (MPR4x0 & MPR5x0) by XBOW [15]. It comprises of a sensor board and a PC/radio board and battery pack, and runs TinyOS [16]. Table 2.2 lists its specification. From the table, we can observe sever limitations in storage, communications range and battery power. Limited CPU and storage resources force code to be small and efficient. Additionally, if we look at the energy consumption, we can see that it takes roughly three times the energy to transmit than that it does to compute. This is based on the assumption that all transmissions are made at maximum power and therefore no range adjustment protocols are in place.

2.3.2 Data Aggregation and Security Concerns

Small sensors such as the above mentioned ones have been used broadly for various applications such as habitat monitoring [3, 20], battlefield surveillance, and structural integrity monitoring on the Golden gate bridge. A large number of such tiny sensors collect mea-

Table 2.2. *XBOW Mote Information [17–19]*

Processor/Radio Board (MOTE)	MPR400CB MPR500CA	MPR410CB MPR510CA	MPR2400CA
Processor Performance			
Processor	16MHz Atmel ATmega128L	16MHz Atmel ATmega128L	16MHz Atmel ATmega128L
Program Flash	128 Kb	128 Kb	128 Kb
Measurement Flash	512 Kb	512 Kb	512 Kb
EEPROM	4 Kb	4 Kb	4 Kb
Current Draw			
Active	8 mA	8 mA	8 mA
Sleep	< 15 uA	< 15 uA	< 15 uA
Multi-Channel Radio			
Center Frequency	868/916 MHz	433 MHz	2400 MHz
Data Rate	38.4 Kbaud	38.4 Kbaud	250 kbps
RF Power	-20 to +5 dBm	-20 to +10 dBm	-24 to 0 dBm
Receive Sensitivty	-98 dBm	-101 dBm	-94 dBm
Outdoor Range	500 ft	1000 ft	300ft
Current Draw			
Transmit	27 mA	25 mA	17.4 mA
Receive	10 mA	8 mA	19.7 mA
Sleep	< 1 uA	< 1 uA	1 uA
Electromechanical			
Battery	2X AA 3V Coin Cell	2X AA 3V Coin Cell	2X AA 3V
External Power	2.7 - 3.3 V	2.7 - 3.3 V	2.7 - 3.3 V

surement data and rely on multi-hop short-range radio communication to send data to the processing center, which is also called base station. The major motivation to use tiny sensors is to save energy consumption and reduce system cost, as recent tiny sensors can work without recharging for several months. The lifetime of sensors, however, depends on effective energy saving strategies such as sensor scheduling and information processing within the network to reduce the data traffic to the base station. One important in-network information processing is data aggregation.

Data aggregation reduces communication costs at the expense of some comparatively cheap processing cycles and increases the overall lifetime of the network. Aggregation may be the calculation of the minimum, maximum, mean, or median of the data. Data compression could be also thought of as a type of aggregation. After data aggregation, the size of data will be reduced significantly, resulting in great energy saving on transmission cost.

As mentioned in Chapter 1, although data aggregation can effectively reduce the amount of data transmitted to the base station, data aggregation provides attackers with good opportunities to inject bogus information into the network. After aggregation, the raw data items are invisible to the base station and thus their authenticity and integrity are hard to guarantee.

2.3.3 Previous Work on Security in WSNs

SIA (Secure Information Aggregation)

SIA [21] by Przydatek et al. provides a protocol to prevent stealthy attacks while allowing for Secure Information Aggregation. This stealthy attack makes an otherwise normally functioning node send bogus information, such as significantly different readings. SIA assumes that there exists a pre-shared key between the aggregator and the home server. Each of the other nodes maintain a key for the home server and their aggregator. The keys are constructed by taking a keyed message authenticating code (MAC) of the nodes ID, thus constructing a unique key for each node based on the common seed. These keys allow for

authentication and confidentiality of the communications.

Furthermore, the protocol follows an Aggregate-Commit-Prove philosophy. The Aggregator calculates the aggregation result using what ever predefined method (mean, median, min, max), then commits to it and proves that it is accurate. The commitment is done by placing all readings at the leaf nodes of a Merkle hash tree, and computing the Merkle hash tree value of the tree, with the root of the tree being the commitment. The Aggregator and Home server then partake in an interactive proof in the Prove stage of the protocol.

This protocol inhibits an Aggregator's ability to change the aggregated value once it has committed to the value. But it does not have any built-in mechanism to prevent a malicious aggregator from utilizing false data to construct the aggregation value and then commit to the fraudulent value.

TinySec

TinyOS as mentioned above is the Operating System that runs on the XBOW motes. The base Operating System does not provide much in the way of security to the communications of the Motes. TinySEC [22] provides various levels of link layer security, including TinySec Authentication and Encryption. The use of Authentication allows users to verify the integrity of the message as well as inhibit non-members from participating in the network; the Encryption provides the confidentiality of the communication.

In order to effectively provide security, TinySec utilizes an Initialization Vector (IV) for the cipher block chaining (CBC) mode based cryptography. The IV allows the system to constantly return a different cipher text despite potentially identical readings. TinySec selects two block ciphers, Skipjack [23] and RC5 [24]. These algorithms were determined to be easily and efficiently implemented within the constraints of the tinyOS and XBOW motes. Skipjack was further favored because it does not require a pre-computed key schedule unlike RC5. Other contenders such as AES, Triple-DES were considered to be too slow in software.

Looking at the results in [22], we come to realize that as much as cryptography is considered a villain and menace to the lifetime of sensor networks with complex calculation,

it also increases the packet size often associated with it.

Resilient Aggregation in Sensor Networks

Wagner shows in [25] that any compromised node can manipulate the end result of the aggregation. He proposes a model that is resilient in the presence of arbitrary changes to a small subset of sensor observations. The focus is on the integrity of the aggregation. A major point of contention is that as we all know the aggregation process may be completely secure and fault free, the actual readings may be fraudulent, and this is the motivation for the statistically resilient aggregation protocol.

To support resilient aggregation, Wagner looks at robust statistics, a field of research that deals with noisy and error-riddled data. Robust statistics is used to tolerate small errors in a large number of measurements and large errors in a small number of measurements. Of the common aggregates calculated, Wagner indicates that Average, Sum, Minimum and Maximum are insecure, in that one or more fraudulent or skewed values can alter the results. For example a higher or lower than expected value can change all of the listed statistical primitives. Though Median is considered to be better than Average, it only takes two compromised nodes before the network can be coerced into providing inaccurate results as well.

Wagner assumes that the aggregation is performed only at the sink node. The statistical methods proposed in [25] cannot be easily implemented when in-network aggregation is used. It is worth noting that in this thesis we propose an in-network protocol where the aggregation is performed by the sensors instead of the powerful sink.

Chapter 3

Secure Routing with the Uncertainty of Intrusion Detection

3.1 Introduction

Mobile Ad-hoc NETworks (MANETs) are networks with mobile nodes without the support of fixed infrastructure. The lack of fixed infrastructure and mobility pose many new challenges. The first major challenge was the routing problem which has been solved for the most part by protocols such as AODV [1] and DSR [2]. The second major challenge is security. Potential deployments of MANETs may be in un-trusted environments. Unfortunately existing security solutions for the Internet are inapplicable to MANETs due to mobility and the lack of a centralized network management point. Security has become an important but hard problem for any realistic applications with MANETs.

The area of MANET security has garnered a lot of attention recently with a host of possible attacks and solutions being published [6, 7, 10, 26]. There are generally two main methods to provide system security: intrusion prevention and intrusion detection. Intrusion prevention usually depends on authentication and cryptographic operations to protect the system. The research in security for the Internet teaches us that intrusion prevention alone may not be enough to provide a comprehensive solution for highly survivable networks. As a result, intrusion detection becomes an indispensable part of a secure system.

Intrusion detection in MANETs, however, tends to have a high false positive rate [27].

The high false positive rate makes it very difficult for accurate intrusion response. This causes a dilemma when using intrusion detection to enhance system security, if the information obtained from the detection module is potentially incorrect. Information that is unreliable will not increase security without high performance penalties.

Nevertheless, the uncertainty of the intrusion detection results can provide us with more or less hints on current network status and can be utilized to help make proper final decisions. Intrusion detection stimulus should be used with a grain of salt, that is, we know it is inaccurate but it can be used as a parameter in the decision making process. Knowing the inaccuracy and dealing with it appropriately is much better than having no information at all.

The main contribution of this chapter is to present a strategy to take advantage of the uncertain knowledge of intrusion detection results. Based on IDS stimulus, the proposed method randomly forwards a routing request to alleviate some of the negative affects the high false alarm rate could have on performance, and to help routing protocols avoid malicious nodes without resorting to complicated cryptographic operations. To the best of our knowledge, we have not seen any research efforts trying to utilize the uncertainty of IDS in secure routing.

The rest of the chapter is organized as follows. In Section 3.2, we introduce a randomized message forwarding mechanism to help routing protocols defend against the rushing attack [7] with the uncertain knowledge from IDS. In Section 3.3, we discuss how the proposed solution addresses the different flavors of the rushing attack. Section 3.4 presents the simulation evaluation of the proposed solution.

3.2 Randomized Message Forwarding Based on Uncertain Knowledge

3.2.1 Assumptions

We believe that security in MANETs can be achieved with the proper combination of intrusion detection and intrusion response. Our proposed solution implements the intrusion response portion of this combination with a few assumptions as to what information the IDS should provide.

The following are the current suggested trace data to be collected for intrusion detection of the rushing attack. This information can easily be obtained from MANET IDS [12].

- Expected number of RREQ packets: it is the average expected number of RREQ packets received for one route during time t.
- Knowledge of neighbors: each node will maintain a table of neighbors. The IDS should be able to provide a confidence level for each neighbor, indicating whether the neighbor is an attacker or has been potentially attacked. For instance, the confidence value of 0% means the neighbor is definitely not an attacker while the confidence value of 100% means the neighbor is surely an attacker.

3.2.2 Randomized Algorithm

The randomized algorithm requires the following parameters:

1. expected number of RREQ packets to be received , n
2. timeout value, t
3. confidence value, cv. (Note that this is assumed to come from the IDS)

The randomized algorithm works as follows. Based on n and cv it generates a random number, i. To generate i we first determine the node space. Node space is essentially the upper bound on the range in which the i can exist. It is determined by multiplying n and

the confidence value, *cv*. In order to enforce randomness, the node space will always have a minimum size of 2. This is to provide an attacker with a maximum probability of 50% to become part of a route. We then use the node space as the modulus of the random number generator.

$$nodespace = max(2, (n * cv)) \tag{3.1}$$

$$i = random()\%nodespace + 1 \tag{3.2}$$

For example, if the confidence level *cv* for the prior hop is 0%, the local IDS agent is confident that the previous hop is not attacking or compromised, so the node space will have a size of 2 determined using (3.1) regardless of the size of *n*. This indicates one of the first two packets that arrive for a given route discovery will be forwarded. On the other hand, if *cv* is 50% and *n* is 10, we will generate a random number *i* using (3.2) with a value between 1 and 5.

Similarly, after each RREQ is received, the associated *cv* value for the sending node is obtained from the IDS. We then use the maximum of the previous *cv* and the new *cv* to create the node space and a new random number. If the number of RREQs is greater than or equal to the value randomly chosen, then the packet at index i is forwarded. This allows the randomized forwarding mechanism to adapt the confidence value and node space as the IDS provides new information. The pseudo code below shows how it all works together.

```
if(not is_neighbor(source) && is_neighbor(previous_hop)){
        //add to a queue for its broadcast id
        rreq_queue *q = add_rreq(bcast_id, source, packet)
        //get confidence
        conf = IDS:get_confidence(previous_hop)
        q->cv = max(b->cv,conf)
        //determine node space
        nodespace = max(2, n*(q->cv/100))
        // calculate the random value
        index_i = (random()%nodespace)+1
        if(q->size >= index_i){
                rreq *r = rids_rreq_get(q, index_i)
                //parse the randomly chosen RREQ
                packet = r->packet
        }else{return}
}//proceed to normal AODV processing of RREQ
```

As discussed in [7], the threshold value t should ideally be based on the number of legitimate hops between the initiator and the node forwarding the RREQ: closer nodes to the source should choose shorter timeouts than far-away nodes. For example, the immediate neighbors of the source should only wait for t, where t is the propagation time from the source to its neighbors. The legitimate number of hops, however, may not be known. Hence we suggest an adaptive method for determining the threshold value as follows:

1. t is initially set to the average propagation time required for one hop.
2. If in time t, we received x packets where x is considerably smaller than n, we simply adjust t as follows: t *=2.
3. We can keep on doing this until either we have received all n packets, or t has reached a maximum T. Since if T is too large it will suffer from severe performance overhead.

The actual value of T should be determined based on the actual network density and traffic. In a dense network a smaller T would be appropriate while in a sparse network T should be larger.

The above algorithm is just an example, demonstrating how the uncertain knowledge from IDS can be utilized for random forwarding. We remind the readers that there possibly exist various types of algorithms that take advantage of the uncertainty of IDS response. For instance, we could add weight to each RREQ packet based on the path length and the associated *cv* value of prior hop to help make wise decisions; we could also use accumulated *cv* values or the maximum *cv* value along the entire path. Since the main purpose of this thesis is to present a framework that integrates the uncertain knowledge in secure routing and thus to trigger more research in this direction, we do not intend to list and analyze all possible algorithms.

3.2.3 Comparison

Superficially, the above proposal seems very similar with the random message forwarding mechanism in [7]. They actually have a lot of differences. One main difference is that the randomized forwarding suggested in [7] waits time t before selecting which RREQ to forward. While our proposal has the potential to forward the very first RREQ received, it could just as easily forward any packet received before timeout t is reached. This is advantageous over waiting for t as it will lessen the delay while still using randomness to increase security. The adaptive backoff allows the routing protocol to adjust to the network status and wait for a larger sample size to choose a RREQ from.

In contrast to the solution in [7], we do not use any cryptography or additional handshaking. This should significantly lower our overhead with respect to their solution, leaving our overhead levels only slightly higher that those of the original protocol. The lower overhead in terms of energy consumption and bandwidth usage is very important in the wireless world. For example, sensors in sensor networks have limited battery life and thus CPU cy-

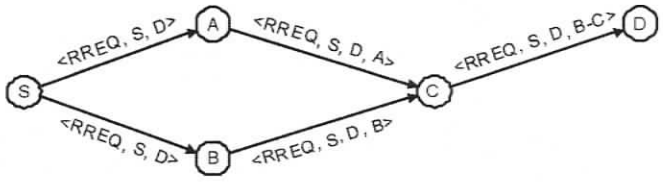


Figure 3.1. *Sample attack scenario*

cles are precious, and transmitting extra routing overhead is not a good use of it.

Our method provides a framework to balance the tradeoff between security and efficiency. The solution proposed here will by no means provide 100% security guarantee, but it does increase security at a low cost computationally, bandwidth wise and in terms of delay. There is no free lunch in secure systems. It is never the case that strong security enforcement requires little performance penalty. In real applications, we are always faced with such balance. A framework such as ours to help balance the tradeoff is invaluable in wireless networks.

3.3 Dealing with Different Types of Rushing Attacks

3.3.1 MAC Layer, Wormhole and High Transmission Power Rushing Attacks

The attacks as previously described in Section 2.2 use some advantage to get RREQs to their destinations quicker by forwarding packets without honoring MAC layer timing constraints, using increased transmission power level, or creating a tunnel with another attacker. The proposed solution protects against these attacks as follows.

Assume the scenario in Figure 3.1 where the sender, S, forwards a packet to nodes A and B, where A is the attacker employing one of the above methods. The attacker will immediately forward the packet to node C, faster than node B.

To protect against the above scenario, we rely on the randomized forwarding mech-

anism. With an appropriate confidence value from the IDS, C randomly chooses which RREQ to forward. In this case the RREQ from A is likely discarded and the RREQ from B is forwarded on since the cv value associated with A is high.

With the wormhole and high transmission power attacks, it can often be the case that no other RREQs will be received within threshold time, t, thereby forcing a node to forward the attacker's RREQ despite the adaptation. Note that the attacker may be selected as the forwarding node with probability of $1/node\ space$ as defined in Formula (3.1). Hence the chances of a successful attack are decreased, and get smaller as the network density increases. The denser the network, the higher the number of neighbors a node has. Since neighbors affect the node space and the size of n, there is a greater chance the attacker will not be selected.

3.3.2 Spurious Packet Flooding

The spurious packet flooding attack, as described in Section 2.2, is typically only effective when cryptography is involved, because of the increased amount of time required to process each RREQ. In normal cases, an attacked node should be able to handle received packets much faster than the speed of wireless transmission from the malicious node. Since we do not require any cryptographic operations, the randomized forwarding mechanism we proposed is not susceptible to this variation of the attack.

3.4 Simulation

3.4.1 Simulation Model

We modify the original AODV protocol to include our proposed randomized forwarding, namely Randomized AODV (RAODV). In this section, we evaluate the performance of the modified AODV protocol via simulation. To truly determine the effectiveness, an adversarial environment was used with four attackers. A normal environment without attackers was

also used for head to head comparison of RAODV and AODV.

We use the ns-2 simulator [28] to evaluate the randomized RREQ forwarding protocol described in Section 3.2. To simulate RAODV, the functionality of processing RREQ messages in AODV was rewritten. We assume the required information is available from an existing IDS.

```
get_confidence(previous_hop, hops)
    if(is_attacker(previous_hop))
        cv = Θ // where Θ is a high value
    else
        cv = f(hops) //where f() is an exponential function
    return cv;
```

A proper IDS would have generated a high confidence value in its analysis of the attackers behavior. In our simulation, the cv value associated with an attacker is above 80%, which we believe achievable for an effective IDS. The RREQs from non attackers were given varying confidence values based on the hop count. This decision is justified because a higher hop count increases the likelihood that an attacker slipped into the path, much like when a message is passed by word of mouth, the further from the source you hear it, the more likely it becomes, that the message has been altered. It is believed that this assumption is realistic because as the length of the path grows, the number of nodes increases and the probability that a node in the path has been compromised also increases.

The attack nodes were modified so they no longer forward anything but routing related packets, acting as a black hole for data. Also, the transmission power was increased to double that of the normal nodes and the RREQ forwarding delay was eliminated. This combination gives the attack nodes a great advantage to successfully perform attacks. In addition to their broadcast enhancements the four attackers were selected specifically to represent the worst possible four nodes in the network to have as attackers. This means that after analysis of the non adversarial scenarios the four selected malicious nodes were considered to be the most likely to be successful attackers. They were chosen based on

distance to source nodes and number of paths they were already partaking in.

A 1500m by 300m rectangular area with 50 nodes was chosen. A random waypoint mobility model was used with a minimum speed of 5m/s and a maximum speed of 10m/s. These parameters were chosen to avoid the problems stated in [29]. The pause times used were 0, 30, 60, 120, 300, 600 and 900 seconds, while the simulation time is 900 seconds in duration [30]. Each simulation scenario was run 10 times. When confidence intervals were calculated for the simulation results, the confidence level was set to 90%.

The channel capacity is set to 2 Mbps. Constant Bit Rate (CBR) traffic was used. There were 32 connections with 37 of the 50 nodes involved using 22 different sources and 25 different destinations. The 32 connections started at random times during the simulation time period. This traffic pattern with the large portion of nodes involved in communication poses heavy traffic in the networks. We intend to evaluate RAODV under these harsh conditions in the following simulation.

3.4.2 Evaluation Metrics

We evaluated control overhead, route creation delay, delivery ratio, the percentage of RREPs forwarded by attackers, and packet drop ratio of the attackers.

Control overhead with and with out the HELLO packets were both calculated for clarity. To get the overhead statistic, the total number of packets sent during simulation was counted and divided by the total number of routing packets sent during the same simulation. This was calculated twice for each simulation, which delivered results with and without the HELLO packets.

Delivery ratio is defined as ratio of the number of data packets sent by the sources over the number of data packets received by the destinations.

The infiltration ratio is defined as the number of times attackers forward RREPs divided by the number of RREPs sent from the sources. This ratio is adequate, however, it does not serve as an accurate measure because it does not take into consideration the amount of time an attacker remains part of a route. So we also looked at the attackers packet drop ratio.

Attacker drop ratio is defined as the ratio of the number of packets dropped by attackers over the total number of packets sent from sources. This metric indicates how many packets were swallowed by the attacking nodes. This metric will let us see the effectiveness of the attackers. The infiltration ratio as described above shows us how many times routes were infiltrated but it does not show how effective the infiltration was.

Route creation delay is very important in on-demand routing. Determining if RAODV drastically increases the route creation delay when compared to standard AODV, is of great importance. For this metric, two routes were randomly chosen: one long (6 hops) and one short (3 hops). The route creation delay times were calculated by taking the difference between the time of the first request sent and the time of the first reply received.

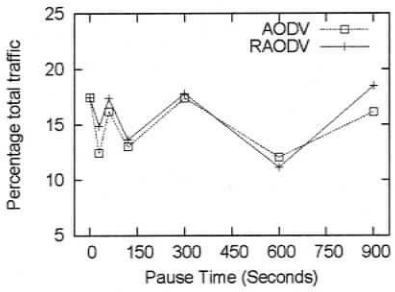
3.4.3 Simulation Results

The results of the simulation are positive with respect to performance. In Table 1, it can be observed that the average AODV response time is very low for the short route but the average RAODV response time is also low. There is only approximately 0.3 seconds difference. The 6 hop route is much different as the difference is approximately 10 seconds, though the median time is lower than that of AODV. From Table 1, RAODV does introduce some delay in route discovery phase. Nevertheless, compared with the results stated in [7] in which the security solution proposed incurs significant increase in route discovery delay, our method is quite acceptable.

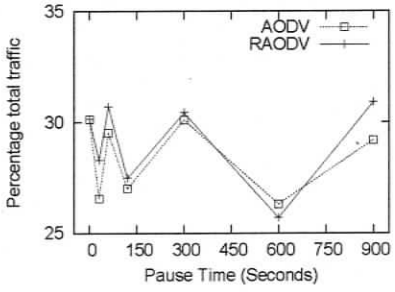
Table 3.1. *Route Creation Delays. Note all delays are in seconds*

Route Length	AODV Mean	RAODV			
		Mean	Median	Standard Deviation	90% Confidence Interval
3 hops	0.4050	0.7485	0.4050	1.0392	0.5405
6 hops	5.0481	15.7657	2.8803	20.1670	10.4898

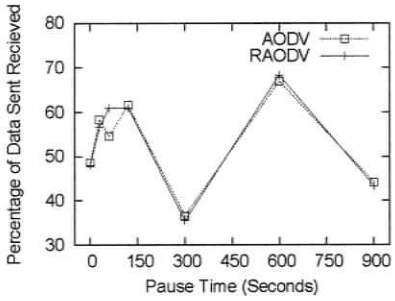
The routing overhead difference between AODV and RADOV is trivial as shown in Figure 3.2(a) and 3.2(b). The overhead levels of both RAODV and AODV follow nearly the same path while both counting the HELLO messages and not counting them. The overhead of RAODV is 15.8% without HELLO messages, while it is 29.1% with them. AODV has an average overhead of 15.0% without HELLO messages, and an average of 28.4% with. From this it could be generalized that RAODV increases the overhead by less than 1% over AODV. This is a big advantage of our proposed solution over other existing solutions in [7].



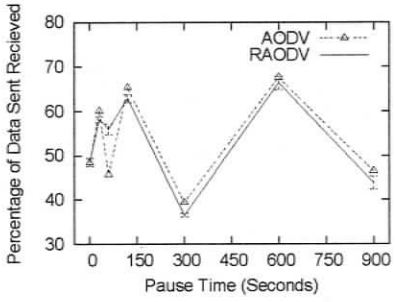
(a) Routing Overhead excluding HELLO



(b) Routing Overhead including HELLO



(c) Data Delivery ratio under non-adversarial conditions



(d) Data Delivery ratio while under attack

Figure 3.2. *Routing overhead and Data delivery ratio at different pause times*

From Figure 3.2(c) and Figure 3.2(d), it becomes obvious that the difference in data delivery ratio is small, just as with overhead. AODV averaged 53.0% delivery while RAODV achieved 53.4% delivery, giving a 0.4% difference in delivery ratio in favor of

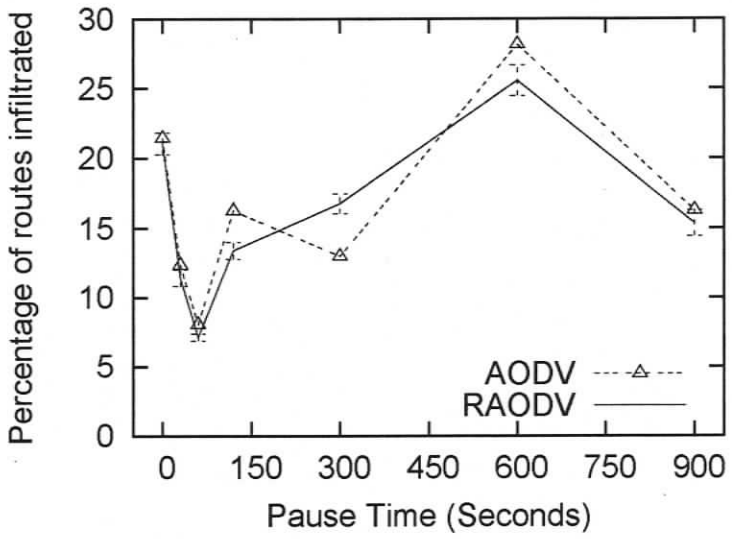


Figure 3.3. *Route infiltration ratio*

RAODV. Intuitively, RAODV should have much higher packet delivery ratio than AODV since RAODV avoids the packet drop by attackers. The similar packet delivery ratios for AODV and RAODV are mainly due to the very harsh and heavy traffic conditions we utilized in our simulations. That is, if RAODV selects alternative paths, the penalty of using the possibly longer length will be enlarged because of heavy end-to-end traffic. Our simulation results actually approximate the worst situation for RAODV or the lowest bound for packet delivery ratio for RAODV. The exaggerated penalty of using long length is confirmed again in Figure 3.4, from which we observe that attackers drop more packets with AODV. The similar delivery ratios of AODV and RAODV imply that more packets are dropped from normal nodes with RAODV because of channel contention caused by heavy traffic through longer paths. The performance of RAODV in light traffic or other "good" network topology (e.g. there are several alternate paths with similar length from a source to a destination) will certainly be better.

For effectiveness with respect to security, route compromise is an important metric to look at. Figure 3.3 shows that in all but one scenario RAODV included attackers less often.

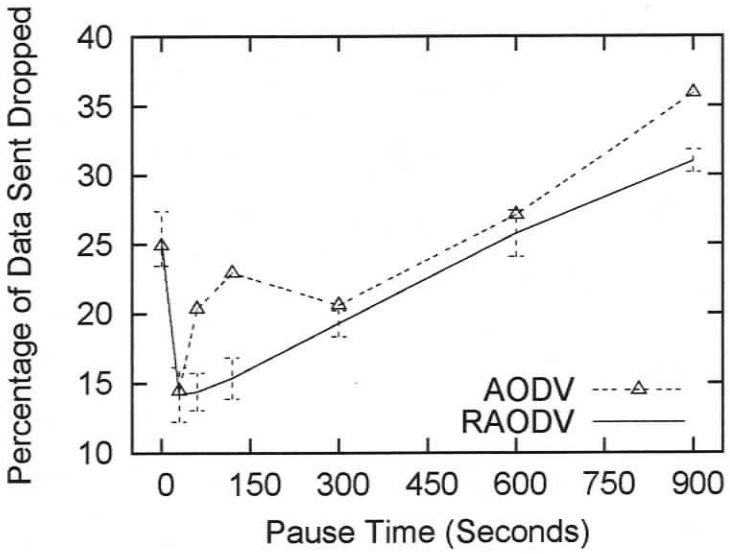


Figure 3.4. *Attacker drop ratio*

For the simulations with lower pause times, we see that both protocols perform similarly. When pause time is long, such as larger than 600 seconds in the simulation, RAODV has smaller infiltration ratios.

This trend is mirrored in Figure 3.4 as we see with RAODV the attackers are dropping fewer packets. Despite the higher infiltration with pause time 300s, we still see RAODV outperforms AODV, indicating that though more routes were infiltrated, the infiltrated paths did not last long. Also, by our definition, the infiltration may not necessarily mean the attacker will be finally included in a valid path. While the attackers exist, AODV dropped 23.8% of the packets sent by the sources, while RAODV only dropped 20.7%, meaning the attackers were 12.4% less effective against RAODV.

To summarize, the randomized forwarding mechanism as simulated only marginally increases control overhead and has similar packet delivery ratio with original AODV even under harsh traffic condition. It does increase security and does limit the effectiveness of the attackers. The results show that a randomized forwarding mechanism removes the predictability that attackers rely on, while only slightly affecting performance.

Chapter 4

Secure Aggregation Without Persistent Cryptographic Operations

4.1 Introduction

Extremely small sensors have been used broadly for various applications such as habitat monitoring, battlefield surveillance, and forest fire monitoring. A large number of tiny sensors collect measurement data and rely on multihop short-range radio communication to send data to the processing center, which is also called the base station or the sink node. The major motivation to use tiny sensors is to minimize energy consumption and reduce system cost, as recent tiny sensors are expected to work without recharging for several months. The lifetime of sensors, however, depends on effective energy saving strategies such as sensor scheduling and in-network information processing to reduce the data traffic to the base station. One important type of in-network processing is data aggregation.

Depending on different application contexts, data aggregation could be in the form of data compression or the calculation of some statistical values, such as the mean, max, and min. While data aggregation can effectively reduce the amount of data transmitted to the base station, raw data items may be invisible to the base station and thus their authenticity and integrity are hard to guarantee. As such, data aggregation is potentially vulnerable to attackers who may inject bogus information or forge aggregated values without being detected. For instance, if a sensor close to the base station is compromised, it may claim a

fake aggregation value to the base station and mislead the base station into trusting the fake information. It may have a disastrous impact if end users respond according to the faulty information.

Some methods have been proposed to solve the above problem [21,25,31–33]. Existing methods depend on complex data authentication operations [31–33] or the statistical features of specific aggregation operations such as the mean, max, and min [25,34,35]. To guarantee correctness, *persistent* authentication operations are used in most existing methods. *Persistent* authentication, though very safe, may consume too much energy and is thus undesirable for sensor networks.

Besides the energy concern, another barrier of using complex cryptographic operations is that the calculation capability of sensor nodes is very limited. For this reason, cryptographic operations may significantly increase the data processing delay and renders the network performance intolerable. For instance, if all data are encrypted and authenticated, the CPU resource for data sampling and digital signal processing will be much reduced, making some CPU-intensive applications, such as audio/video monitoring, extremely hard to implement.

In this thesis, we assume the existence of a secure mechanism but use it only when necessary, i.e., only when cheating activities are detected. This strategy is fundamentally different from existing solutions in that its security is based on cheating detection instead of persistent data authentication. This strategy can reduce energy consumption and more importantly can save the CPU resource. It, however, requires the support of a lightweight and scalable cheating detection method, which is a quite challenging task.

We are motivated to solve this problem and have made the following contributions. First, it introduces some topological constraints when building a Secure Aggregation Tree (SAT), which facilitates the monitoring of the behavior of each aggregation sensor node. Second, when the aggregated values from an aggregation node are in doubt, a weighted voting scheme is proposed to decide finally whether the aggregation node is properly behaving or is cheating. Third, if a misbehaving node is detected, a local recovery scheme

is presented to re-build SAT so that the misbehaving node is excluded from the aggregation tree. Finally, analysis and simulation are performed to demonstrate the effectiveness of SAT. Since no cryptographic operations are required when all nodes work honestly, our method is lightweight. Since no centralized operations are needed at the base station, our method also scales very well.

The rest of the chapter is organized as follows. In Section 4.2, we introduce the assumptions on which our method is based. In Section 4.3, we describe the details of building a SAT. Section 4.4 presents the mechanisms of cheating detection and local recovery. In Section 4.5, we discuss the criterion in setting the confidence value to the detection of a cheating node. Section 4.6 and Section 4.7 present the analytical study and the simulation results respectively.

4.2 Assumptions and Attack Model

Two sensor nodes are called one-hop neighbors if they can communicate directly with each other and two-hop neighbors if they can communicate via two-hop radio transmission. In this paper, neighbors by default are referred to one-hop neighbors unless we explicitly indicate two-hop neighbors.

We assume that *a node cannot impersonate its neighbors*. This is not a strong assumption since if a node, A, is in the promiscuous listening mode, it can quickly detect any transmission from its neighboring nodes with A's identity. This assumption is to exclude the possibility that a node impersonates its father node in the aggregation tree to send false aggregated data such that an honest father is voted out of the network.

We assume that two neighboring sensor nodes have mechanisms for secure communication so that they can decrypt and authenticate each other's messages. There are a lot of key distribution methods proposed for sensor networks that can meet this requirement [22, 36–38]. This seemly strong assumption does not mean that our method will always use message encryption and authentication. In contrast, they are used *only when*

cheating behavior is detected. If all sensor nodes work honestly, no cryptographic operations are required.

Attacking or malicious nodes are assumed to be randomly selected, and randomly distributed within the network. These malicious nodes are assumed to be working independently of each other, conducting their attacks without cooperation from additional compromised nodes in the network.

In this chapter, we focus on the integrity and authenticity of data. A secure sensor network must be able to effectively defend against an adversary from injecting bogus information into the network, since such attacks are very harmful and could mislead the network operator into making bad decisions and render the monitoring systems useless.

4.3 Building a Secure Aggregation Tree (SAT)

4.3.1 The structure of SAT

Our Secure Aggregation Tree (SAT) is motivated by the work of *watchdog* [39]. If we could build an aggregation tree such that any child node can monitor the behavior of its father node, then the cheating activities of any non-leaf (aggregation) nodes can be detected. In this paper we do not consider the cheating behavior of leaf nodes since it is indistinguishable between cheating and malfunctioning for a leaf node. In order, to allow a child node to monitor its father node's behavior, it is required that the child node should be able to know all the messages from its sibling nodes to the father node. In other words, a father node together with its children nodes should form a *clique*.

An example of such an aggregation tree is shown in Figure 4.1, where both the solid lines and the dotted lines between two nodes indicate that the two nodes can hear each other, and the solid lines represent the aggregation tree.

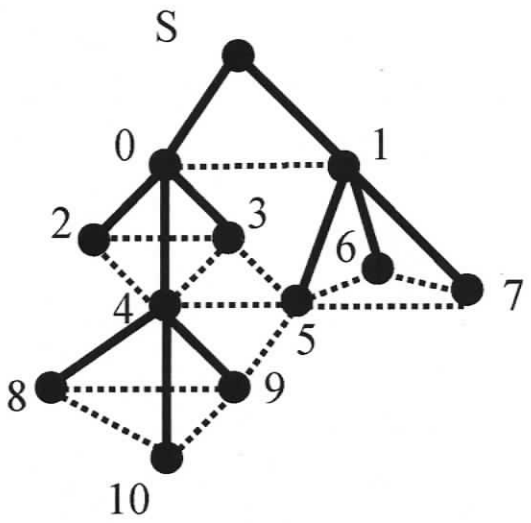


Figure 4.1. *An example of the secure aggregation tree*

4.3.2 A Distributed Algorithm to build SAT

In this section, we propose a distributed algorithm to build SAT with the assumption that each node knows its one-hop and two-hop neighbors. The one-hop neighbors can be easily found with beacon messages, and the information of two-hop neighbors can be found with one local broadcast from each sensor node, indicating who are its one-hop neighbors.

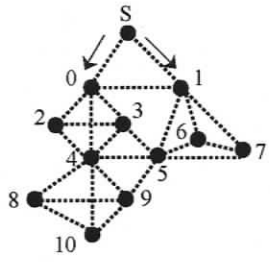
The distributed algorithm builds the aggregation tree beginning from the sink node and includes four steps as follows.

1. Step 1: The sink node locally broadcasts an *invitation* message to all of its one-hop neighbors, indicating that they should be its children. The *invitation* message includes the IDs of all nodes that a father node wants to invite to join the aggregation tree as its children. It should also include the hop count value to make a node aware of its minimal hop count to the sink node. The hop count value in the *invitation* message from the sink node is set to zero. A node sets its initial hop count value to infinity (i.e., a large integer like 10000) and updates its hop count value as one plus the minimal hop count value in all received *invitation* messages.

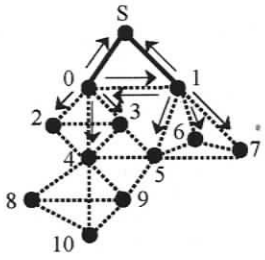
2. Step 2: Once a node receives an *invitation* message, if this node has not joined the aggregation tree and the *invitation* message includes this node as a child node, then this node joins the aggregation tree and records the sender of the *invitation* message as its father node. It locally broadcasts a *join* message to notify its neighbors about this decision. This *invitation* message is also called *activating invitation* message since it enforces the node to join the aggregation tree. Once a node joins the tree, subsequently received *invitation* messages will be recorded for future use if the hop count value in the *invitation* messages is smaller than the node's current hop count value. More specifically, these *invitation* messages will be used to select a candidate aggregation node if a current working aggregation node is compromised, as described in detail in Section 4.4.3. To avoid forming a poor aggregation tree, an additional rule should be applied, although this rule is rarely used if the network density is high: if a node receives an invitation message but the hop count value included in the message is 2 hops larger than its current hop count value, then this invitation message is ignored. We call this rule the *optimization rule* since it excludes the possibility of creating a poor aggregation tree that requires large energy consumption on data delivery.
3. Step 3: After a node joins the aggregation tree, by checking its one-hop and two-hop neighbors, excluding those indicated in the *activating invitation* message, it finds all the cliques that it belongs to. If such cliques cannot be found, then this node works as a leaf node. Otherwise, it selects the maximal clique and locally broadcasts an *invitation* message with the hop count value increased by one, indicating that all other nodes in the selected clique should be its children.
4. Step 4: Repeat step 2 and step 3 until all non-isolated nodes have joined the tree.

Note that if a node is disconnected from the sink node, it will not receive any *invitation* message and will not join the tree. In this case, the node is an isolated node and cannot be used by any means.

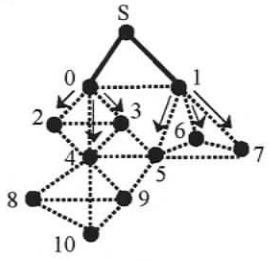
As an example, Figure 4.2 illustrates how an aggregation tree is built step by step.



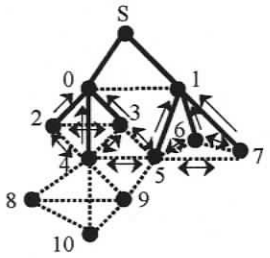
(a) S broadcasts an *invitation* message to nodes 0 and 1.



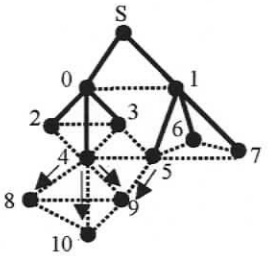
(b) Nodes 0 and 1 join the aggregation tree and broadcast a *join* message.



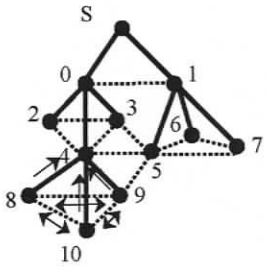
(c) Node 0 finds the maximal clique {0, 2, 3, 4} and invites nodes 2, 3, 4 to join the aggregation tree. Similarly, Node 1 finds the maximal clique {1, 5, 6, 7} and invites nodes 5, 6, 7 to join the aggregation tree.



(d) Nodes 2, 3, and 4 broadcast *join* messages to indicate node 0 as their father. Similarly, nodes 5, 6, and 7 broadcast *join* messages to indicate node 1 as their father.



(c) Node 4 finds the maximal clique {4, 8, 9, 10} and invites nodes 8, 9, 10 to join the aggregation tree. Similarly, Node 5 finds the maximal clique {5, 9} and invites node 9 to join the aggregation tree.



(e) Nodes 8, 9, and 10 broadcast *join* messages to indicate node 4 as their father. Node 9 records the *invitation* message from node 5 for future use if we assume that the *invitation* message from node 4 arrives first.

Figure 4.2. *An illustration of the steps in building up the SAT*

Due to the topological constraint that an aggregation node together with its children should form a clique, it is possible that some nodes may not join the aggregation tree even if they have paths to the sink node. We call such nodes *sparse nodes* since they have only sparse set of neighboring nodes. Nevertheless, as demonstrated later, the ratio of the number of sparse nodes over the total number of sensor nodes is *extremely small* if the network density is reasonably high[1]. As such, we could simply require the sparse nodes to send their messages to the sink node without performing any in-network processing. Any secure routing and secure unicast mechanisms could be applied to the messages sent from the sparse nodes.

It is possible that in Step 2 and Step 3 local broadcast messages may collide and the correct information may not be received by receivers. Fortunately, this problem can be easily avoided in our tree buildup process. If sensor nodes are roughly time synchronized, the order of the broadcast messages from the children nodes can be scheduled by the father node. For instance, when a father node makes the selection of its children, it can arrange an arbitrary order for the children nodes broadcasts and piggyback this information in the *invitation* message. Each child node is permitted to broadcast only in the allocated timeslot. Furthermore, to reduce broadcast overhead, a node may combine the *join* message and the *invitation* message into one single broadcast.

At the end of the SAT buildup process, each node should record the following information:

1. the IDs of its one-hop and two-hop neighboring nodes,
2. the ID of its father node in the aggregation tree,
3. the IDs of its sibling nodes in the aggregation tree.

Note that the IDs of a node's sibling nodes in the aggregation tree are obtained from the local broadcast of the *join* messages from the sibling nodes.

[1]Reasonably high degree in this case is considered to be greater than 5 neighbours.

4.4 Cheating Detection for Data Aggregation

4.4.1 Detection of Potential Misbehavior

The cheating detection is very similar to the *watchdog* introduced in [39], where each node works in the promiscuously listening mode to monitor all transmissions within its maximal radio range. With *watchdog*, each node, after sending a packet to its next hop node, listens to the channel to check if its next hop node relays the packet correctly. Similarly, in our method, due to the topological constraints in SAT, each node can hear all messages sent to its father node and can monitor the message sent from its father node to its grandfather node to check if the father node performs data aggregation correctly. If a node's father node sends out a value that is *significantly* different from a correct aggregation value, the node will raise an *alert*. The criterion of setting a proper threshold value for raising an alert will be discussed in Section 4.5. In this paper, we differentiate an *alert* message from a *detection-confirmation* message. An *alert* message means a potential compromise, while a *detection-confirmation* message indicates the final confirmation of the existence of a cheating node.

Ideally, if a sensor node can hear all messages sent to its father and track the values that have been aggregated, the above cheating detection will be accurate. In practice, however, it is possible that some messages to the father node are lost or the father node may not use exactly the same set of values for aggregation if time is not synchronized very well. In both cases, cheating detection with SAT may generate false alarms. The false alarm rate is obviously dependant on the specific application context and the criterion of raising alerts, which will be disclosed in Section 4.5.

To draw a consistent conclusion about an aggregation node's behavior, we assign each *alert* with a confidence value. This confidence value is calculated based on the specific aggregation function and is used to indicate the likelihood that the aggregation node is cheating. In the next section, we propose a weighted voting method to make the final decision regarding whether or not an aggregation node is cheating.

4.4.2 Weighted Voting

First of all, we want to remark that all control messages in the weighted voting process and the local recovery process should be encrypted and authenticated with some underlying secure communication mechanisms. This is to guarantee the correct final decision once potential misbehavior is detected. Also, we stress again that *the secure communication is required only when an aggregation node is working improperly*.

Once a sensor node detects that its father node might be cheating, it sends out an *alert* message to all its neighbors except the father node. To keep a compromised node from sending out fake alert messages and also to keep the voting process invisible to the father node, the alert message should be encrypted and authenticated. Note that the voting process should be secure, since otherwise an attacker can compromise a node and paralyze its father node as well by sending out fake alerts. It may be necessary to make the voting process invisible to the father node, since a smart attacker may temporarily recover the compromised node to normal operations just to escape the detection once he/she realizes a weighted voting process about his/her cheating behavior is going on.

An *alert* message should include the following information:

1. The cheating node's ID,
2. The detecting node's ID,
3. The confidence value of the alert.

The confidence value indicates the likelihood of a correct detection. A large confidence value indicates that the aggregation node is more likely to be cheating.

Once receiving *alert* messages from neighboring nodes, a node first checks if the cheating node is its father node. If yes, it calculates the weighted confidence value using the following formula:

$$F = \frac{\sum_{i=1}^{m_1} f_i}{m}$$

where f_i is the confidence value included in the alert message from the sibling node i, m is the total number of sibling nodes, and m_1 is the number of sibling nodes that send out an

alert message.

If the weighted confidence value F is larger than a predefined threshold value, the father node is assumed to be cheating, and a *detection-confirmation* message will be broadcast within a given hop limit. Any node receiving the *detection-confirmation* message will use the following local recovery mechanism to avoid using the compromised node.

We stress again that we assume the availability of an underlying secure mechanism to support secure message transmission and broadcast so that we do not need to worry about the integrity and authenticity of the *alert* and the alarm messages. If such an underlying secure mechanism does not exist, the security of the sensor network when facing attacks cannot be guaranteed in any ways. Also, note that a node cannot receive the *detection-confirmation* message only if it is separated from the network or it is separated from the network by the compromised node. In both cases, we have no means to use that node again.

4.4.3 Local Recovery

If a sensor node, A, receives a *detection-confirmation* message, it checks if the cheating node is one of its child nodes or its father. If the cheating node is one of its child nodes, A will ignore any messages from that child.

If the cheating node is its father node, A will select another candidate father from its recorded *invitation* messages. Note that deadlocks cannot be formed since a node only records *invitation* messages that have smaller hop count values than its current hop count value. In other words, sensor node A selects another neighboring sensor node, B, as its father node *only if* B has a smaller hop count value to the sink node than A. This is to avoid the deadlock situation where two sibling nodes select each other as the potential father node. If a candidate father node cannot be found, then the current node becomes an isolated sensor node since none of its fathers can be used for reliable aggregation.

The newly formed "isolated" sensor nodes actually have a physical path to the sink node, although the path has to include some misbehaving nodes and may not be secure. If the sensor network still requires the sensing data from these "isolated" sensor nodes,

more secure mechanisms such as data signature and authentication must be utilized for each message from the newly formed "isolated" nodes.

4.5 Examples of Raising alerts

4.5.1 Problem Modeling

Obviously, the selection of a proper threshold value to decide whether a father node is cheating relies on the application context and the aggregation function. *There is no unified criterion suitable for all scenarios.* In this section, we introduce the criterion on selecting the proper threshold value for several broadly used aggregation functions, namely, mean, max, and min. We stress that the methods introduced in this paper are *purely for exemplary demonstration* that cheating detection is feasible. Since the detection rate and the false alarm rate depend on specific application requirements, it is desirable to design new criterion for specific application contexts.

There are mainly two reasons that the aggregated value at a father node is different from that calculated at a child node. The first is that some packets sent to the father node may be lost, and the other is that the father node uses a slightly different set of values for aggregation due to time asynchrony. In our model, a father node knows the number of its child nodes. When an aggregation calculation is required, the father node will use the most recently received values from its child nodes as the input. That is, *we do not assume that data transmissions are well synchronized* so that a child node knows exactly the data aggregated in the father node.

For instance, as shown in Figure 4.3, the child nodes, $C_1, C_2, \ldots, C_m$, send data packets to the father node, F. Assume that during a time period each child node $C_i (i = 1, \ldots, m)$ sends n packets to the father node, denoted as $D(C_i, 1), D(C_i, 2), \ldots, D(C_i, n)$ respectively. Due to packet losses or time asynchrony, the value of the aggregation function calculated by the father node might be any value in the set

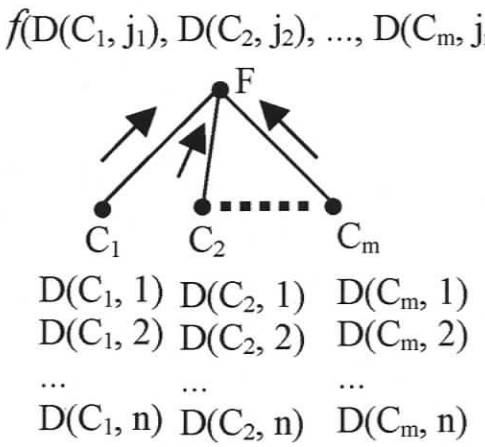


Figure 4.3. *An illustration of the data aggregation model*

$\{f(D(C_1, j_1), D(C_2, j_2), D(C_3, j_3), \ldots, D(C_m, j_m))\}$ where f is the aggregation function and $j_k, k = 1, 2, \ldots, m$ might be any value in $\{1, 2, \ldots, n\}$.

For a child node, when it monitors its father's behavior, it calculates the same aggregation function

$$f(\overline{D(C_1, j_1)}, \overline{D(C_2, j_2)}, \overline{D(C_3, j_3)}, \ldots, \overline{D(C_m, j_m)}))$$

where $\overline{D(C_i, j_k)}(i = 1, \ldots, m)$ is the newest data received from the sibling node i. As a reasonable assumption, we assume that $|D(C_i, j_k) - \overline{D(C_i, j_k)}|$ is bounded by δ, since it is unlikely that during a short time period readings from the same sensor change dramatically [2]. We also assume that data from different child nodes are independent [3].

Our problem could then be modeled as follows. Given an aggregation function f, an arbitrary positive value λ, a set of values $D_1, D_2, \ldots, D_m$, and another set of values $\overline{D_1}, \overline{D_2}, \ldots, \overline{D_m}$ with each $\overline{D_i}$ randomly selected from $[D_i - \delta, D_i + \delta](i = 1, 2, \ldots, m)$ respectively, what is the probability that $|f(D_1, D_2, \ldots, D_m) - f(\overline{D_1}, \overline{D_2}, \ldots, \overline{D_m})| \geq \lambda)$? We denote the above probability as P_c.

[2] This phenomenon is also called temporal correlation of sensory data.

[3] With this assumption, we assume that the spatial correlation of data is negligible. Refer to Section 4.5.5 for further discussion.

The value of λ indicates the absolute difference between the aggregated value calculated at a father node and that monitored at a child node. Given a small threshold value θ, if $P_c \leq \theta$, then the child node should raise an alert since it is unlikely (i.e., P_c is too small) that the father node should generate such a different aggregated value. The confidence value for the alert is set to $1 - P_c$.

4.5.2 Mean

Denote the mean of $D_1, D_2, \ldots, D_m$ as $S = \frac{\sum_{i=1}^{m} D_i}{m}$. Also denote the mean of $\overline{D_1}, \overline{D_2}, \ldots, \overline{D_m}$ as $\overline{S} = \frac{\sum_{i=1}^{m} \overline{D_i}}{m}$, where $\overline{D_i}$ is randomly selected from $[D_i - \delta, D_i + \delta]$ respectively. Note that S is a constant and $\overline{S}$ is a random variable. We are asked to estimate $P_c = Pr(|\overline{S} - S| \geq \lambda)$.

It is obvious that when $\lambda > S + \delta$ or $\lambda < S - \delta$, $P_c = 0$.

When $S - \lambda \leq S \leq S + \lambda$, by Chebyshev's bounds we have

$$P_c \leq \frac{VAR[\overline{S}]}{\lambda^2}.$$

Since $\overline{D_i}, i = 1, 2, \ldots, m$ are independent, it is easy to calculate that

$$\begin{aligned} VAR[\overline{S}] &= \frac{1}{m}\sum_{j=1}^{m} VAR[\overline{D_i}] \\ &= \frac{1}{m}\sum_{j=1}^{m}(E[\overline{D_j}^2] - (E[\overline{D_j}])^2) \\ &= \frac{1}{m}\sum_{j=1}^{m}(\frac{1}{2\delta}\int_{D_j-\delta}^{D_j+\delta} \overline{D_j}^2 d(\overline{D_j}) - D_j^2) \\ &= \frac{1}{3}\delta^2 \end{aligned}$$

In conclusion,

$$P_c \begin{cases} = 0 & \text{if } \lambda > S + \delta \text{ or } \lambda < S - \delta \\ \leq \frac{\delta^2}{3\lambda^2} & \text{otherwise} \end{cases}$$

4.5.3 Min

Denote the *min* of $D_1, D_2, \ldots, D_m$ as $m = min\{D_1, D_2, \ldots, D_m\}$. Also denote the *min* of $\overline{D_1}, \overline{D_2}, \ldots, \overline{D_m}$ as $\overline{m} = min\{\overline{D_1}, \overline{D_2}, \ldots, \overline{D_m}\}$, where $\overline{D_i}$ is randomly selected from $[D_i - \delta, D_i + \delta]$ respectively. Note that m is a constant and $\overline{m}$ is a random variable. We are asked to estimate $P_c = Pr(|\overline{m} - m| \geq \lambda)$.

For the function of *min*, we can get an accurate formula instead of a bound to calculate the above probability. First, let us check under what situation the above probability is zero. It is easy to see that it is impossible for $\overline{m}$ to get a value smaller than $m - \delta$ or larger than $m + \delta$ if the aggregation node is honest. That is, if the aggregation node is honest, the inequality $-\delta \leq \lambda \leq \delta$ must be true. Therefore, we get if $\lambda > \delta, P_c = Pr(|\overline{m} - m| \geq \lambda) = 0$. In other words, if the absolute difference between the aggregated value calculated at the father node and that monitored at a child node is larger than δ, the child node is sure that the father node is cheating and raises an alert with 100% confidence.

Next, we calculate the probability $Pr(|\overline{m} - m| \geq \lambda)$ when $\lambda \leq \delta$.

For each random variable $\overline{D_i}, i = 1, \ldots, m$, the *cdf* (cumulative distribution function) of $\overline{D_i}$ is

$$F_i(x) = Pr(D_i < x) = \frac{x - D_i + \delta}{2\delta}.$$

Therefore, the *cdf* of $\overline{m}$ is

$$\begin{aligned} F(x) &= 1 - \prod_{i=1}^{m}(1 - F_i(x)) \\ &= 1 - \prod_{i=1}^{m}(1 - \frac{x - D_i + \delta}{2\delta}). \end{aligned}$$

By calculation,

$$
\begin{aligned}
P_c &= Pr(|\overline{m} - m| \geq \lambda) \\
&= Pr(\overline{m} - m \geq \lambda) + Pr(\overline{m} - m \leq -\lambda) \\
&= 1 - F(m + \lambda) + F(m - \lambda) \\
&= 1 + \prod_{i=1}^{m} \frac{\delta + D_i - m - \lambda}{2\delta} - \prod_{i=1}^{m} \frac{\delta + D_i - m + \lambda}{2\delta}.
\end{aligned}
$$

In conclusion,

$$
P_c = \begin{cases} 0 & \text{if } \lambda > \delta \\ 1 + \prod_{i=1}^{m} \frac{\delta + D_i - m - \lambda}{2\delta} - \prod_{i=1}^{m} \frac{\delta + D_i - m + \lambda}{2\delta} & \text{otherwise} \end{cases}
$$

4.5.4 Max

Denote the *max* of $D_1, D_2, \ldots, D_m$ as $M = max\{D_1, D_2, \ldots, D_m\}$. Also denote the *max* of $\overline{D_1}, \overline{D_2}, \ldots, \overline{D_m}$ as $\overline{M} = max\{\overline{D_1}, \overline{D_2}, \ldots, \overline{D_m}\}$, where $\overline{D_i}$ is randomly selected from $[D_i - \delta, D_i + \delta]$ respectively. Note that M is a constant and $\overline{M}$ is a random variable. The calculation of $P_c = Pr(|\overline{M} - M| \geq \lambda)$ is the same as that when the aggregation function is *min* in the previous section.

4.5.5 Further Discussion

The analysis in previous sections is purely for demonstration purpose that cheating detection is feasible. For this reason, we have made certain assumptions that (1) an aggregation node does not have data itself and (2) data from different children node are independent or their correlation is negligible. While these assumptions hold true for some applications, it is not surprising that readers can always find counter-examples. It is unrealistic to propose a generic solution workable in all situations.

If an aggregation node itself also has data to add in the aggregation function, the only way to detect whether an aggregation node is a cheating node is to require the aggregation

node to piggyback its own data together with the aggregation value. Since we use clique-tree for aggregation, this add-on value is known by all children such that cheating detection can be performed.

If the spatial correlation in sensory data is not negligible, the above analytical results do not hold anymore and must be adjusted. Since the spatial correlation is usually positive correlation [4], the actually bound will be looser. For instance, the variation of the mean function will be larger than that calculated above due to positive correlation. Nevertheless, it is impossible to quantify the correlation without considering dataset from a specific application. To summarize, accurate cheating detection criteria must consider specific application context, which is obviously beyond the focus of this paper.

4.6 Cost Analysis

4.6.1 Rationale

The rationale to use SAT is that energy could be saved by requiring no cryptographic operations when sensor nodes behave correctly. This rationale is justifiable only if (a) the energy cost on the monitoring with SAT in a long run is smaller than the energy cost on cryptographic operations and (b) SAT does not incur larger energy cost on data transmission than other types of aggregation tree. In the following, we demonstrate that the two conditions hold true for SAT.

4.6.2 Energy Cost on Monitoring

In order to check if the energy cost of monitoring is smaller than traditional cryptographic operations, we compare the energy cost on computing aggregation functions, e.g., max, min, and mean, and that on a widely used cryptographic algorithm in sensor networks, RC5.

[4] Data from different children usually have the same trend (increasing or decreasing).

The amount of computational energy consumed by a function on a given microprocessor is primarily determined by the power consumption of the processor, the clock frequency of the processor, and the number of clocks needed by the processor to compute the function. We assume that energy consumption cannot be significantly reduced via a reduction in clock frequency. Since the function and the efficiency of its software implementation determine the number of clocks necessary to perform the function, the processing overhead in terms of the *Number of Basic Operations* can reflect the implementation efficiency and the energy consumption on computing a function.

Thus, we calculate the *Number of Basic Operations* of the aggregation functions and compare them with those of RC5. We consider Addition, XOR, Shift (1 bit), Fetch (fetch a value from the main memory to a register), and Store (store a value in a register to the main memory) as our basic operations. In particular, we choose RC5-32/12/X, (i.e., 32 bits words, 12 rounds, and X as the key length) based on the algorithm in [24]. The key-expansion routine is the most time-consuming part in running the RC5 algorithm. Because most wireless sensor networks that adopt RC5 as their underlying cipher require the S-Table to be computed in advance to speedup their sensors operation, we do not consider the cost of computing the S-Table in our analysis.

The cost of performing one general n-bits multiplication is roughly as $\frac{n}{2}$ additions and $\frac{n}{2}$ shifts in average on n-bit registers. Since a division can be reduced to a multiplication, we use the same estimation for the division. Also, the same estimation is made to the general modulo. A multiplication by 2 is a left-shift operation; the operation of $(n \mod 32)$ is considered one XOR operation (in fact, we need a bitwise AND). In Rivest's algorithm, "shift B bits ($\lll B$)" means "shift $(B \mod 32)$ bits." Thus, there is a bitwise AND involved. Also, we use 16 as the average value of $(B \mod 32)$. For RC5, we assume that the values of A and B, the two words to be encrypted, remain in the registers during the course of computation.

The breakdown of the number of RC5's basic operations is illustrated in Table 4.1. Here we do not count the computation overhead of the S-Table's pre-computation.

Table 4.1. *Numbers of Basic Operations in RC5-32/12/X.*

Operation	8-bit Processor 8 byte block	8-bit Processor 16 byte block	8-bit Processor 32 byte block
Addition	400	800	1600
XOR	384	768	1536
Shift	3072	6144	12288
Fetch	320	640	1280
Store	16	32	64
Total	4192	8384	16768

To make the comparison plausible, we assume that the input data size to the aggregation function is the same as the input data size to RC5. The actual cost of the aggregation function should be smaller than that we calculate here, since RC5 is a block cipher that enforces the input data padded into blocks when the input data size is not divisible by the size of the block. The number of basic operations for calculating min/max and for calculating mean is listed in Table 4.2 and Table 4.3 respectively. Note that when we calculate the min/max among n data items, we require $n - 1$ comparison operations and the energy cost on each comparison operation is considered equal to that of one Addition.

Tables 4.1, 4.2, and 4.3 illustrate the breakdown of basic operations of calculating RC5, min/max, and mean respectively. Combine these three tables together, we obtain Table 4.4, which depicts the comparison of the number of basic operations. It is obvious that aggregation operations are much simpler than cryptographic operations. This fact clearly justifies the energy efficiency of using SAT compared to using persistent cryptographic operations.

Table 4.2. *Numbers of Basic Operations in Calculating Min/Max.*

Operation	8-bit Processor 8 byte block	8-bit Processor 16 byte block	8-bit Processor 32 byte block
Addition	7	15	31
XOR	0	0	0
Shift	0	0	0
Fetch	8	16	32
Store	8	16	32
Total	23	47	95

Table 4.3. *Numbers of Basic Operations in Calculating Mean.*

Operation	8-bit Processor 8 byte block	8-bit Processor 16 byte block	8-bit Processor 32 byte block
Addition	11	19	35
XOR	0	0	0
Shift	4	4	4
Fetch	9	17	33
Store	9	17	33
Total	33	57	105

Table 4.4. *Numbers of Basic Operations in RC5-32/12/X and Aggregation Functions.*

	8-bit Processor 8 byte Packet	8-bit Processor 16 byte Packet	8-bit Processor 32 byte Packet
RC5-32/12/X	4192	8384	16768
Min/Max	23	47	95
Mean	33	57	105

4.6.3 Energy Cost on Data Transmission

4.6.3.1 A note on Radio Transmission

The structure of SAT fully takes advantage of the broadcast feature of radio channels. Since a node together with its sibling nodes and its father node forms a clique, no extra energy is required for receiving packets from sibling nodes and the father node if the node is set to promiscuous listening mode. This is the same as the *watchdog* in [39].

4.6.3.2 Aggregation Returning Values with a Fixed Data Size

We assume that the energy cost on data transmission is mainly decided on the amount of data transmitted. Although it is true that the transmission distance can also impact the energy cost, we ignore this impact since the radio range of sensor nodes is generally very small. Within a short range, the difference in the energy cost with different transmission distances may be negligible.

For most data aggregation functions such as min, max, and mean, the energy cost with different aggregation trees is roughly the same. This is because these aggregation functions return a constant number of bits no matter what the number of the input data items is. In this case, each sensor node, no matter if it is a leaf node or an aggregation node, sends out the same amount of bits and thus has roughly the same energy consumption on data transmission. The energy cost on data transmission with any aggregation tree is $(n-1)*e$ where n is the number of total sensor nodes and e is the energy cost on data transmission at each sensor nodes.

4.6.3.3 Aggregation Returning Values with Variable Data Sizes

For some aggregation functions, the return value may not have constant number of bits. Instead, the size of the return value is a function of the total size of input data. For example, data compression is this type of aggregation function. Under this situation, we are concerned whether or not SAT may incur extra data transmission overhead compared with

other types of aggregation trees.

Note that data compression may not be an effective in-network data processing method since all information has to been processed in the sink node finally. Due to this reason, our analysis should be considered only as a guideline of cost estimation for *potential data aggregation functions* that may exist in the future. This is also the reason that we do not perform analysis for compression-like aggregation in the previous section.

We assume an aggregation function that compresses the input data with a compression ratio of $\gamma, 0 < \gamma \leq 1$. We assume that all sensor nodes generate the same amount of measurement data, and each sensor node will aggregate the data reported from its children (if any) and the data generated by itself.

Without losing generality, we base our calculation on the assumption that each sensor generates 1 byte of measurement data. For a given aggregation tree, denote the number of leaf nodes as l_0, and denote the number of the fathers of the leaf nodes as l_1, and so on. Denote the total layers of the aggregation tree as L and the total number of nodes as n. Then the total transmission cost of an aggregation tree is:

$$\begin{aligned} C &= \sum_{i=1}^{L} \sum_{j=1}^{i} \gamma^j l_i \\ &\geq \gamma(n-1) \end{aligned} \tag{4.1}$$

where $l_1 + l_2 + \ldots + l_L + 1 = n$.

Theorem 1 If the aggregation function compresses the input data with a constant compression ratio $\gamma(0 < \gamma \leq 1)$, the transmission cost of SAT is bounded by a constant approximate ratio to that of the optimal aggregation tree.

Proof: Given a sensor node, A, assume that all paths between A and the sink node, S, are listed as $P_0, P_1, \ldots, P_m$ in the increasing order of the path length. Assume that the length of P_0 is l. Our proof is based on the fact that the optimal aggregation tree with the minimal communication cost must include A as its layer-l node (The root of the tree is layer-0, and the children of the root is layer-1, and so on). This is because the cost

for transmitting one byte from A is $\sum_{j=1}^{i} \gamma^j$ where i is the number of layer to which A belongs and $\gamma(0 < \gamma \leq 1)$ is the compression ratio. This is a monotonic function with i, and thus the higher the layer to which A belongs, the larger the communication cost of the aggregation tree.

If we assume that each local broadcast requires the same time, then the first received invitation message (not necessarily an activating invitation message) should come along the shortest path from the sink node. Assume that the father node of A along the path P_0 is B (There may exist multiple paths with the same path length as P_0, but we consider only one path for simplicity). According to the tree buildup process of SAT, the following possibilities exist:

1. Case a: If A belongs to the maximal clique checked at node B, then A will receive an *activating invitation* message and join SAT. In this case, the cost of delivering messages from A with SAT will be the same as that with the optimal aggregation tree.
2. Case b: If A does not belong to the maximal clique checked at node B, then A is either a sparse node, which is a rare case from later simulation when network density is reasonably high, or A becomes a grandchild of B if one of B's child invites A as its child. A cannot become B's grand-grandchild since only two-hop neighbors are considered when building SAT and also because of the *optimization rule*, i.e., if a node receives an activating invitation but the hop count value included in the message is 2 hops larger than its current hop count value, then this invitation message is ignored. In this case, the cost of delivering one byte from A to the sink node is $\gamma^{l+1}(0 < \gamma \leq 1)$ more bytes compared to the optimal aggregation tree.
3. Case c: A does not become a descendant of B, but instead it becomes a child along other path $P_i, i > 0$. According to the *optimization rule*, the cost of delivering one byte from A to the sink node requires at most $\gamma^{l+2} + \gamma^{l+1}(0 < \gamma \leq 1)$ more bytes compared to the optimal aggregation tree.

Therefore, according to Formula 4.1, in the worst case the ratio of energy cost of SAT (excluding the sparse nodes) over the energy cost of the optimal aggregation tree is

$$\begin{aligned} & \frac{\sum_{i=1}^{L}\sum_{j=1}^{i}(\gamma^{j+2}+\gamma^{j+1}+\gamma^{j})l_i}{C} \\ &= \frac{C+\sum_{i=1}^{L}\sum_{j=1}^{i}(\gamma^{j+2}+\gamma^{j+1})l_i}{C} \\ &\leq 1+\frac{\gamma^2(\gamma+1)\sum_{i=1}^{L}l_i}{C} \\ &\leq 1+\gamma^2+\gamma. \end{aligned} \tag{4.2}$$

4.7 Simulation Study

4.7.1 Simulation Model

In this section, we perform simulation study to further demonstrate the feasibility and the effectiveness of SAT. The simulator was implemented in JAVA. We want to illustrate that SAT does not introduce extra transmission cost compared to other commonly used tree structures and the proportion of the sparse nodes in SAT is extremely small for dense networks. In addition, we want to justify that the cost for local recovery is practically acceptable.

For comparison purposes, we implemented SAT and other types of tree structures, namely, BFT (Breadth-First Tree) and BFT-D (Breadth-First Tree with the Distance constraint). In BFT-D, we list node A as node B's child during the breadth-first search only when node A is further away from the sink than node B. The simulation was run on a variety of scenarios. The sensor nodes were deployed randomly in a 1000 meters by 1000 meters area with nodes having a transmission range of 50 meters. We changed the number of sensor nodes from 500 to 5000 to change network density and placed the sink node in four different locations, i.e., the up-left corner $(0, 0)$, the top-middle $(500, 0)$, the left-middle $(0, 500)$, and the center $(500, 500)$. In order to get convincing results, we ran each scenario with 3 different random seeds. The simulation results were obtained by calculating

the average of all runs.

Three performance metrics were considered:

1. The average depth of leaf nodes. It is calculated as the average hop count of all leaf nodes to the sink node. This metric is used to evaluate the average height of the aggregation tree and thus to estimate the cost for data transmissions with the tree.
2. The ratio of the sparse nodes. It is calculated as the ratio of the number of sparse nodes (refer to Section 4.3.2) over the total number of sensor nodes. Since sparse nodes cannot use SAT for their message transmission, a small number of sparse nodes is desirable.
3. The number of local candidate paths. It is calculated as the number of alternative paths from a node to its SAT grandparent node, i.e., paths without using the node's SAT father. Such paths could be used in local recovery (Section 4.4.3) and thus are called local candidate paths. We searched for the local candidate paths with limited number of hops.

Note that we do not simulate the false alarm rate and the detection rate in this thesis because of three reasons. First, in Section 4.5 we have already provided the criterion for raising alerts, and the probability of false alarms has been mathematically modeled. Second, the test results of false alarms and detection rate with synthetic sensing data may be too good to be realistic. Data from realistic applications are required for this evaluation. Third, there is no generic criterion suitable for all application scenarios. The test of false alarms and detection rate has to be application specific, which is beyond the focus of this thesis.

4.7.2 Simulation Results

4.7.2.1 The average depth of leaf nodes

Figure 4.4 shows results of the average depth of the leaf nodes with different tree structures. From the results, the three tree structures, SAT, BFT, and BFT-D, have very close

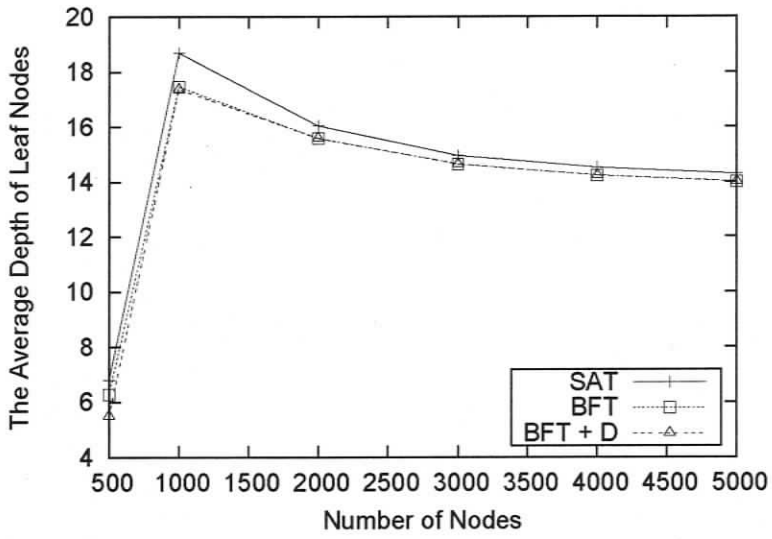


Figure 4.4. *Comparison of different tree structures*

performance. The average depth of leaf nodes indicates the average height of the aggregation tree and thus the approximate cost on message delivery with the aggregation tree. Figure 4.4 illustrates that SAT, despite its advantage for aggregation monitoring, does not incur obvious extra overhead on message delivery compared to other types of trees. This is because the *optimization rule* in SAT (refer to Section 4.3.2) excludes the possibility of creating a poor aggregation tree, and also because the inherent feature of radio broadcast enhances the chance of forming a clique structure locally. Another phenomenon is that the average depth of the leaf nodes increases when the number of nodes is increased from 500 to 1000, and then it remains roughly stable with further increase of the number of nodes. This is because the network connectivity is poor when the number of nodes is smaller than 1000. When the network density is high enough, for instance, when the number of network nodes is larger than 2000, each node can find the shortest path to the sink node with the length roughly equivalent to the direct distance between the node to the sink node.

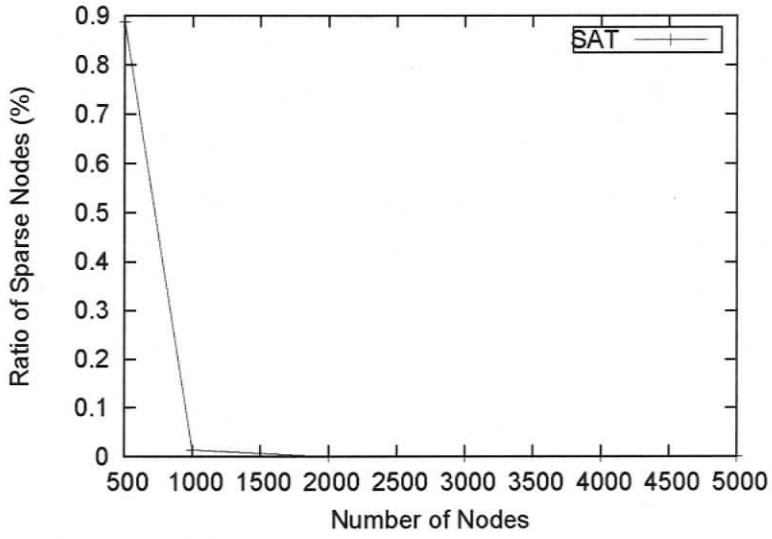


Figure 4.5. *The ratio of sparse nodes*

4.7.2.2 The ratio of sparse nodes

Figure 4.5 shows the ratio of the number of sparse nodes over the total number of nodes. From the figure, we can see that when the network is sparse, e.g., when the number of nodes is smaller than 1000, the number of sparse nodes is not negligible. Nevertheless, when network density becomes higher, the performance of SAT becomes significantly better with the ratio of sparse nodes quickly dropping to almost zero. This phenomenon justifies our previous claim that with dense networks, the cost on message delivery from the sparse nodes is negligible.

4.7.2.3 The number of local candidate paths

Table 4.5 shows the number of candidate paths from a node to its grandfather node without using its current father node on the SAT. Such candidate paths could be used for local recovery once an aggregation node is detected to behave abnormally. From the figure, with a looser constraint on the hop count from a node to its grandparent, more candidate paths

Table 4.5. *Number of Candidate Paths*

Nodes	1 Hop	2 Hops	3 Hops	4 Hops
500	0.058997	0.572271	3.277286	18.79056
1000	0.022493	0.761523	7.954369	77.62297
2000	0.003250	0.889625	23.00842	444.9818
3000	0.002028	0.933139	45.53275	1323.001
4000	0.001917	0.937104	75.66433	2975.243
5000	0.001933	0.960550	113.3157	5620.514

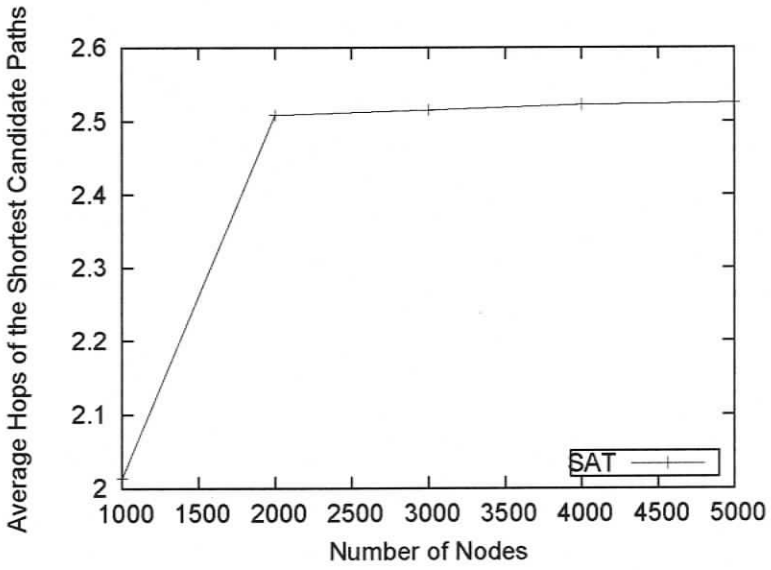


Figure 4.6. *Average hops of the shortest candidate paths*

could be found and thus the chance of successful local recovery is higher. But a candidate path with larger hop count requires more message transmissions. There is a tradeoff between the possibility of local recovery and the recovery cost. We observe from the figure that with a hop count constraint of 3, more than 15 local candidate paths could be found in most cases. Also, from Figure 4.6, the average hops of the shortest path from a node to its grandfather node without using its current father node is smaller than 3. This clearly demonstrates that the energy cost for local recovery is acceptable since local broadcast with hop limit of 3 suffices for the requirement in dense networks.

Chapter 5

Conclusions and Future Work

5.1 Summary

This thesis aimed at proposing effective and lightweight security enhancements to infrastructureless wireless networks. Two types of services, namely MANET routing and WSN data aggregation, are investigated. This thesis presents two new ideas that accomplish the security goal without requiring large overhead. One idea is to use the uncertainty of intrusion detection to enhance routing security, the other is to build an aggregation tree with special topology constraints to facilitate behavior monitoring.

5.2 Utilizing the Uncertainty of Intrusion Detection to Enhance Routing Security

In this thesis, we presented a scheme that utilizes the uncertainty of IDS results to enhance network security. We did not intend to provide a 100% security guarantee, a fundamental design principle for other existing solutions [7]. Instead, we propose a framework to effectively balance the tradeoff between security and performance efficiency, a case we are actually faced in realistic applications. Such tradeoff mechanisms are extremely important in wireless networks because of stringent constraints of bandwidth and energy resources.

From recent research activities in intrusion detection for MANETs [12, 27], IDS tends

to raise high false positive alarms. Controlling false positive alarms is very challenging for MANETs, and this difficulty is unlikely to be solved in a short time. It is a discouraging fact that very few researchers are willing to "waste" energy in such an elusive problem. This thesis tries to put some fresh air in this direction by showing that even the uncertain knowledge of IDS can still be utilized to enhance security. We have observed that predictability is an enemy of security and inaccurate intrusion detection is an enemy of correct intrusion response. We contend that the randomization we proposed provides a possible solution to both problems, giving increased security with performance comparable to that of AODV as specified in [1].

In the future, we plan to propose other mechanisms in utilizing the uncertainty of IDS and perform analysis on the bounds of randomized algorithms in making correct decisions.

5.3 Secure Aggregation Without Persistent Cryptographic Operations

Traditional solutions to secure data aggregation use persistent data authentication that incurs heavy computational overheads even if no attackers or misbehaving nodes exist in the network. Such computational overheads waste energy and may greatly reduce the available CPU resource for data processing. At the worst case, some CPU-intensive applications, such as audio/video monitoring, may become impossible. An ideal security solution should not perform any cryptographic operations when the network works correctly, and uses data authentication only when misbehaving nodes are detected. For this purpose, we propose a Secure Aggregation Tree (SAT) with which it is very easy to observe the behavior of aggregation nodes without resorting to persistent data authentication. With analysis and simulation, we demonstrate the feasibility and effectiveness of using SAT to monitor the aggregation operations.

Any security solution cannot be perfect, and SAT has no exception. Although the frame-

work of using SAT to monitor misbehavior is feasible in general, we have found that in some specific application scenarios, new algorithms are required to make SAT effective. Future work includes:

1. In this thesis, we only consider the temporal correlation of sensing data but do not consider the spatial correlation of sensing data in the analysis of raising alerts. If the spatial correlation of sensing data could be modeled as well, then the detection accuracy could be improved.
2. In this thesis, we only consider the *clique-tree* topology for aggregation monitoring. It is interesting to see if other types of aggregation topology could be used for this purpose.

Bibliography

[1] C. Perkins, E. Belding-Royer, and S. Das, "Ad-hoc on-demand distance vector routing, rfc 3561," July 2003.

[2] D. Johnson, D. Maltz, and Y. Hu, "The dynamic source routing protocol for mobile ad-hoc networks," April 2003.

[3] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Application driven systems research: Habitat monitoring with sensor networks," in *Communications of the ACM Special Issue on Sensor Networks.*, June 2004.

[4] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Mobile Computing and Networking*, 2000, pp. 56–67.

[5] Y. Huang, W. Fan, W. Lee, and P. S. Yu, "Cross-feature analysis for detecting ad-hoc routing anomalies," in *Proceedings of the 23rd International Conference on Distributed Computing Systems*, May 2003.

[6] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless networks," in *Proceedings of IEEE Infocomm 2003*, April 2003.

[7] ——, "Rushing attacks and defense in wireless ad hoc network routing protocols," in *Proceedings of the ACM Workshop on Wireless Security (WiSe 2003)*, Sept. 2003.

[8] ——, "Ariadne: A secure on-demand routing protocol for ad hoc networks," in *Proceedings of the Eighth ACM International Conference on Mobile Computing and Networking (Mobicom 2002)*, Sept. 2002.

[9] A. Perrig, R. Canetti, J. Tygar, and D. Song, "The tesla broadcast authentication protocol," *RSA CryptoBytes*, vol. 5, no. Summer, 2002.

[10] Y.-C. Hu, D. B. Johnson, and A. Perrig, "Sead: Secure efficient distance vector routing for mobile wireless ad hoc networks," *Ad Hoc Networks*, vol. 1, no. 1, pp. 175–192, 2003.

[11] T. Clausen and P. Jacquet, "Optimized link state routing protocol, rfc 3626," October 2003.

[12] Y. Zhang, W. Lee, and Y. Huang, "Intrusion detection techniques for mobile wireless

networks," *ACM - Kluwer Wireless Networks Journal*, vol. 9, no. 5, pp. 545 – 556, September 2003.

[13] W. W. Cohen, "Fast effective rule induction," in *Machine Learning: the 12th International Conference*. Morgan Kaufmann, 1995.

[14] T. Joachims, *Making large-scale SVM learning practical*. MIT-Press, 1999.

[15] CrossbowTechnology, "Crossbow mica 2 mote," 2004. [Online]. Available: http://www.xbow.com

[16] Berkley, "Tinyos," 2004. [Online]. Available: http://www.tinyos.net

[17] CrossbowTechnology, "Mica2 datasheet 6020-0042-06 rev c," 2005. [Online]. Available: http://www.xbow.com/Products/Product%5Fpdf%5Ffiles/Wireless%5Fpdf/MICA2%5FDatasheet.pdf

[18] ——, "Mica2dot datasheet 6020-0043-04 rev d," 2005. [Online]. Available: http://www.xbow.com/Products/Product%5Fpdf%5Ffiles/Wireless%5Fpdf/MICA2DOT%5FDatasheet.pdf

[19] ——, "Micaz datasheet 6020-0060-02 rev b," 2005. [Online]. Available: http://www.xbow.com/Products/Product%5Fpdf%5Ffiles/Wireless%5Fpdf/MICAz%5FDatasheet.pdf

[20] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler, "An analysis of a large scale habitat monitoring application," in *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 3-5 2004.

[21] B. Przydatek, D. Song, and A. Perrig, "Sia: Secure information aggregation in sensor networks," in *ACM SenSys 2003*, Nov 2003.

[22] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link layer security architecture for wireless sensor networks," in *Second ACM Conference on Embedded Networked Sensor Systems (SensSys 2004)*, November 2004.

[23] NSA, "Skipjack and kea algorithm specifications," May 1998. [Online]. Available: http://csrc.nist.gov/CryptoToolkit/skipjack/skipjack.pdf

[24] R. Rivest, "The rc5 encryption algorithm," in *Proceedings of the 1st Workshop on Fast Software Encryption*, 1995, pp. 86–96.

[25] D. Wagner, "Resilient aggregation in sensor networks," in *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04)*, October 2004.

[26] P. Ning and K. Sun, "How to misuse aodv: A case study of insider attacks against mobile ad-hoc routing protocols," in *IEEE Information Assurance Workshop*, 2003.

[27] K. Wu and B. Sun, *Intrusion detection for wireless mobile ad hoc networks*, Y. X. Y. Pan, Ed. Nova Science Publishers, 2004.

[28] "The network simulator ns-2 version 2.27," March 2004. [Online]. Available: http://www.isi.edu/nsnam/ns

[29] J. Yoo, M. Liu, and B. Noble, "Random waypoint considered harmful," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, April 2003, pp. 1312 – 1321.

[30] S. R. Das, C. E. Perkins, and E. M. Royer, "Performance comparison of two on-demand routing protocols for ad hoc networks," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, March 2000, pp. 3 – 12.

[31] H. Cam, S. Ozdemir, P. Nair, and D. Muthuavinashiappan, "Espda: Energy-efficient and secure pattern-based data aggregation for wireless sensor networks," in *IEEE Sensors 2003 Conference*, Toronto, Canada, October 22-24 2003.

[32] L. Hu and D. Evans, "Secure aggregation for wireless networks," Jan 2003.

[33] W. Zhang and G. Cao, "Group rekeying for filtering false data in sensor networks: A predistribution and local collaboration based approach," in *IEEE INFOCOM*, March 2005.

[34] A. Boulis, S. Ganeriwal, and M. B. Srivastava, "Aggregation in sensor networks: an energy - accuracy tradeoff," *Elsevier Ad-hoc Networks Journal (special issue on sensor network protocols and applications)*, 2003.

[35] 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*. ACM Press, 2004, pp. 239–249.

[36] A. Perrig, "The biba one-time signature and broadcast authentication protocol," in *Proceedings of the Eighth ACM Conference on Computer and Communications Security (CCS-8)*, Nov. 2001, pp. 28–37.

[37] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "Spins: Security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, Sept. 2002.

[38] R. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, and P. Kruus, "Tinypk: securing sensor networks with public key technology," in *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*. ACM Press, 2004, pp. 59–64.

[39] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Mobile Computing and Networking*, 2000, pp. 255–265.