A Case Study of Wireless Sensor Network Attacks

Siebe Datema

Delft University of Technology

A Case Study of Wireless Sensor Network Attacks

Master's Thesis in Computer Science

Parallel and Distributed Systems Group
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology

Siebe Datema

September 22th, 2005

Author
Siebe Datema

Title
A Case Study of Wireless Sensor Network Attacks

MSc presentation
September 30th, 2005

Graduation Committee

Prof. dr. ir. H. J. Sips (chair)	Delft University of Technology
Dr. K. G. Langendoen	Delft University of Technology
Dr. ir. A. Lo	Delft University of Technology

Abstract

This thesis describes the research that was done on the security aspects of Wireless Sensor Networks. Different network layers were examined for their potential shortcomings and weaknesses for a variety of known attacks. Different attacks were performed on real hardware and in simulations to study their usefulness and effectiveness on a running wireless sensor network. The resources such as battery power and memory, which are very valuable in wireless sensor networks, that are needed by the attacker will be examined, as well as how many nodes or what part of the network can be affected.

Preface

The work lying before you is my Master of Science thesis, which contains a description of my research conducted from September 2004 until September 2005 at the Parallel and Distributed Systems Group, Faculty EEMCS, Delft University of Technology. I have always been intrigued by hackers, cryptanalysts, and security experts and their efforts to find faults or weaknesses in systems. When offered the opportunity to do similar research, but on the subject of wireless sensor networks, I decided this was something on which I would like to work.

Before continuing with my thesis, I would like to thank a few people who have helped me during my Master's project. First and foremost, I would like to thank Katelijne, for her help and encouragement. I would like to thank my parents for their support during my years at the university. My last thanks go to the many people who shared their opinion with me during many off and on topic discussions.

Siebe Datema

Delft, The Netherlands,
September 22th, 2005

Contents

Chapter 1

Introduction

A wireless sensor network (WSN) consists of a number of small devices, usually equipped with sensors, that together form a network that can perform tasks by communicating with each other using a radio. Nowadays, WSNs are becoming more commonplace and can be found in research projects and commercial applications as well as defence projects. The use of these WSNs can range from intrusion detection through production line monitoring to wildlife observation, and people are finding new and better ways to use the capabilities of WSNs every day. With these new applications, it is easy to forget the security aspect of the WSN.

Even when a WSN is used for a peaceful application, such as detecting the presence of animals, it should be protected adequately. Some see it as a sport to try and find weaknesses, exploits, and bugs in everything that is hard- and software. In common computer software, such as Internet browsers and operating systems, people are constantly searching for security flaws. Some will report them to get them fixed while others use them for their own benefit. There are already a host of known ways to take advantage of insecure wireless networks based on Wi-Fi technology. With WSNs becoming more popular and mainstream, it is only a matter of time before people will try to find ways to exploit the WSNs deployed in their neighbourhood. This is similar to what is happening now with War Driving [1], where people drive around with a laptop with a wireless interface trying to find unprotected wireless home or office networks. Figure 1.1 shows the wireless networks in Delft that have been found in this way.

If one does not want its WSN to be a test subject or the victim of vandalism, the security aspect should not be overlooked. Therefore, it is useful to know how today's WSN software and protocols hold up against a variety of attacks and how likely it is that these attacks really take place. Because of this, the thesis question is: "How secure are the different network layers in wireless sensor networks?"

In this thesis, we study the different network layers, from the OSI model (see Figure 1.2), in WSNs for their resilience against attacks. It is important to see how much effect an attack has on the communication between different parts of the network and the network's ability to recover or work around it. The energy

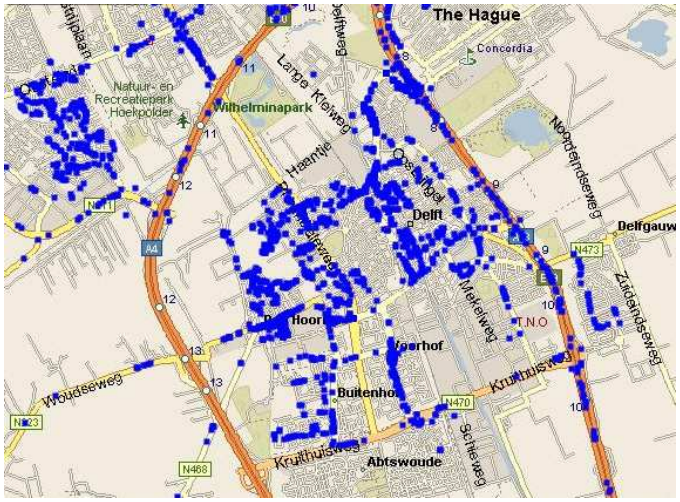


Figure 1.1. Wireless networks (in blue) found in Delft.

	OSI
7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Data link
1	Physical

Figure 1.2. The OSI model stack.

and processing resources that are required for such an attack are examined as well, because if all the energy of the attacker is used up before the attack produces the required result, its usefulness is questionable. When dealing with a completely unknown WSN, the number of attacks that can be performed is limited. Therefore, it is valuable to know how much information about the opponent is needed to be able to perform a certain attack. During this Master's project only the most commonly used software is used, but most of the results can be applied to other systems and algorithms as well.

The remainder of this thesis contains six chapters and their contents is as follows. Chapter 2 contains related work. In Chapter 3, the used software and hardware as well as their set up are described. In Chapters 4, 5, and 6, attacks on the physical, link, and routing layer are described, respectively. In Chapter 7, the achieved results will be discussed. In Chapter 8, conclusions and future work are described.

Chapter 2

Related Work

When one wants to have different electronic devices to communicate with each other, a network will have to be set up to offer a way for the different devices to send each other messages. When such a network is installed in for example an office building using cables, also called guided media, it is very difficult to eavesdrop on the communication from the outside. If spying on the communication is really necessary, one would need to go into the building and install spying equipment or try to break in through an outside connection into the network. However, if a wireless network is installed, it is most likely that someone with a suitable receiver can receive most of the communication while standing outside the building [2, 11]. Therefore, the security aspects of wireless networks should not be overlooked.

There are still enough individuals and even companies who do not secure their Wi-Fi networks at all or not enough. The encryption standard used with a lot of Wi-Fi networks is Wired Equivalent Privacy or WEP, which has been shown to be easily breakable [19]. But also the newer and more secure Wi-Fi Protected Access (WPA) protocol can be insecure when not properly configured [15]. This goes to show that when security of wireless networks is concerned, one should always be on guard.

The area of WSNs is not as big as that of wireless home and office LANs using Wi-Fi, but the security risks are still the same or even bigger because it is tempting to trade security for prolonged battery life. Wi-Fi based networks have less of an energy, size, and cost constraint and therefore, the extra energy consuming calculations, needed for most security measures, are less of an issue. Wi-Fi network cards can even incorporate dedicated encryption hardware, which, if placed on a WSN node, would only add to its cost and size.

Because of these considerations different proposals have been made to deal with the security issues. The most well known and widely used is TinySec [9], which provides encryption at the link layer. TinySec can be used with TinyOS [3]. TinyOS is an open-source operating-system specifically designed for use with WSNs and is the most widely used development platform. TinySec uses a shared key for its encryption, which means all nodes in the network share the same key. TinySec

is claimed to have around 10% energy overhead.

Other encryption methods that are being studied focus on two areas. The first area is the reduction of computation and memory overhead without sacrificing the level of security [16]. But because most encryption methods rely on the usage of large numbers and multiple rounds of data manipulation and calculations that try to hide the input, little progress has been made. A second area is the use of public/private keys instead of shared keys to obtain better security for the network, but these implementations also incur a big performance hit. Solutions proposed, such as described in [21], try to put as much of the calculation workload on other devices besides the nodes, such as a laptop-class base station. Another possible solution proposed is the use of elliptic curves [14]. The results still show a big memory footprint and a large number of calculations.

Because of the drawbacks of encryption many WSNs still send their messages in the clear. This will make the WSN vulnerable for attacks on the network and routing layer. An overview of possible WSN attacks is given in [10, 23]. A lot of the attacks described are derived from attacks on wired and Wi-Fi networks but adapted for WSNs. Most of the attacks can be very disruptive if the WSN is not protected. A list of the attacks mentioned and the possible defence, or countermeasures, can be seen in the table below. Explanations of the attacks will be given in the relevant chapters.

Layer	Attack	Countermeasure
physical	jamming	frequency hopping
link	collisions	error correcting codes
network and routing	Selective forwarding	redundancy
	Sinkhole	authorisation
	HELLO Flood	link verification
	Sybil	authentication
	routing cycles	routing checks

TinyOS is a well known operating system for use with WSN hardware and probably used the most in projects and research. In [10], attacks are given that should be theoretically possible in TinyOS. How the attacks should work is explained, but they are not tested in real-life and it remains to be seen if, for example, routing cycles can really be created. In [23], the possibilities of several attacks on WSNs are discussed and the need to take security into account at design time. They argue that adding security to existing protocols often focuses on using cryptographic techniques, but when building a protocol from the ground up one can take different attacks into account and build in resistance against them.

Routing protocols that take security into account have been proposed, such as [6] and [7]. The INSENS [6] protocol relies on authentication by the base station. If nodes need to send messages to nodes other than the base station, these messages will have to be sent through the base station for authentication. This can be very inefficient if one wants to have neighbours communicate with each other. In [7], resilience against attacks is achieved by setting up multiple routes between the

same source and destination. If one of these routes fails due to an attack or other problems, messages can still be sent through another route.

From the above it is clear that WSNs can not go without adequate security. Unfortunately there has not been much practical research towards the security of today's WSNs. This thesis will remedy that by performing different attacks on these WSNs and study their results.

Chapter 3

Test Set-Up

To study the security of the different network layers, attacks known from literature were performed using real nodes and simulations. In this chapter an overview is given of how the tests were set up. In Section 3.1 and 3.2, the used software and hardware are discussed, respectively. In Sections 3.3, the set-up of the experiments is discussed. In Chapter 4, 5, and 6 we will perform experiments using the set-up on the physical layer, link layer, and network layer respectively.

3.1 Software

As discussed in Chapter 2, most of the projects dealing with WSNs use TinyOS as their operating system. This is not the only operating system available for sensor networks, but it is one of the most widely used open source operating systems available. Therefore, we have chosen TinyOS as the operating system to work with. TinyOS is an event-driven operating system designed for sensor network nodes that have very limited resources. TinyOS and programs for TinyOS are written in NesC [8]. The NesC programming language is designed specifically for TinyOS and it is based upon the concept of components that are connected or wired together to form a program. This should make it easier to change components, such as for example a routing algorithm or sensor module. This is seldom done, however.

For the TinyOS operating system, several routing protocols have been written. Some of these are adaptations of existing wireless ad-hoc routing protocols, such as Destination-Sequenced Distance-Vector Routing, while others are written specifically for TinyOS, such as Multihop. There are not many implemented routing protocols beyond the well known protocols present in the TinyOS CVS repository. The routing algorithms that will be examined in Chapter 6 are:

- Multihop
- Mintroute

- DSDV Hop Count
- DSDV Quality
- TinyAODV

3.2 Hardware

For a number of experiments, wireless sensor nodes were used. The nodes need to be able to run the software mentioned in Section 3.1 and also be able to offer Received Signal Strength Indication (RSSI) values. RSSI values are a measurement of the received signal strength from a node's radio and are needed to detect communication during some of the tests.

The nodes that are both readily available to us, and also fit the above mentioned requirements are the MICA2 and MICA2DOT from Crossbow, shown in Figures 3.1 and 3.2, respectively. There are four nodes of both types available for the experiments. Because the MICA2s are easier to work with, due to their simpler connection to a PC, they are used instead of the MICA2DOTs.

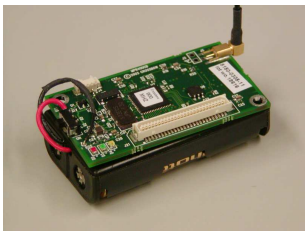


Figure 3.1. A MICA2 node.

Figure 3.2. A MICA2DOT.

3.3 Set up

A short explanation of the experiments and the used test set-up can be found in the following subsections.

3.3.1 Physical Tests

The real world tests with the MICA2 nodes are mainly used for the physical layer and link layer attacks, where the physical properties associated with radio communication are a factor in the effectiveness of an attack. The tests are done with some of the MICA2 nodes running a sniffer program to gather data, to perform jamming and to cause collisions. To obtain accurate results, all factors during the tests have to remain the same, because only a slight change in the location of the nodes or objects in the vicinity can have a large impact on radio reception characteristics.

The RSSI values that are received from the radio are actually ADC (analog-to-digital converter) values that first have to be converted to dBm. This is done with the following calculation (where V_{bat} is the voltage of the battery):

$$V_{rssi} = V_{bat} \cdot \frac{ADC}{1024}$$
$$RSSI(dBm) = -50 \cdot V_{rssi} - 45.5$$

3.3.2 Simulations

Tests with higher layers, such as the network and routing layers, are performed with simulation software. This is done to avoid having to set up a complete network that is suitable for routing layer tests. With a real network, it is very time consuming to change the network topology and also test the effectiveness of an attack. This is because on the one hand the network is being disrupted, while on the other the nodes need to be able to indicate what is happening, which are opposite goals. One does not have this problem in simulations where all nodes can be monitored.

The simulator that is used is TOSSIM [12], which stands for TinyOS Simulator. It is included with TinyOS together with a program called TinyViz, that can be used to visualise the WSN network running in the simulator and also process debug data from some or all of the nodes. To use it, a TinyOS application needs to be compiled specifically for the simulator. The compiled executable can then be started with command line arguments telling the simulator how many nodes to simulate, what radio model and topology to use and its debugging and visualisation settings. The simulations will either start running immediately or if specified, wait for TinyViz to connect to it. TinyViz can then show which node is sending messages to other nodes, who is broadcasting and which LEDs on the nodes are on and off.

One drawback of the simulator is that all simulated nodes run the same application. This is a disadvantage when one of the nodes needs to act as a node that is performing an attack. When testing an application using a certain routing algorithm, the source code has to be changed to allow one of the nodes to become an attacker when starting up.

During the experiments, different topologies are used. For some experiments, a special topology will be constructed to better show how an attack affects the network. When testing routing protocols in a WSN, the topology in Figure 3.3 will often be used. The dots represent the nodes while the edges represent neighbour relations between nodes. Only nodes that share an edge can communicate with each other. The base station will be node zero, located in the upper left corner of the topology. The communication will be without errors unless indicated otherwise.

When testing attacks on the routing protocols in simulations in Chapter 6, TinySec will not be used.

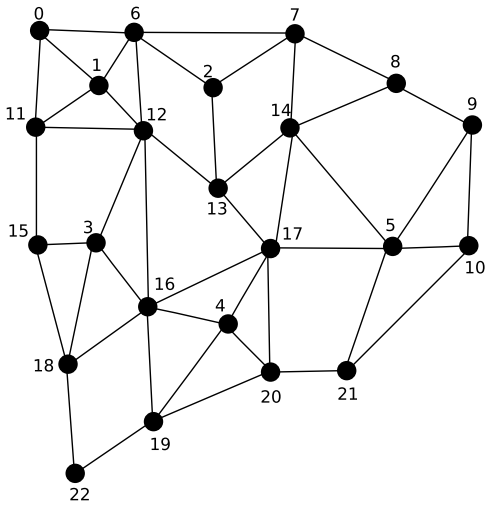


Figure 3.3. One of the topologies used during experiments.

Chapter 4

Physical Layer

In this chapter, the attacks on the physical layer are described in detail. In Section 4.1, a brief introduction to radio communication is given. In Section 4.2, the set-up and results of various jamming tests are discussed.

4.1 Radio Communication

The most common way of communication between nodes in a WSN is by using a radio. A radio propagates signals through the air which contain the data that it is sending. Another radio can receive these signals and retrieve the sent information. When two radio transmitters are sending signals at the same time and a radio receiver is within radio range of both, it is possible that the signals interfere and nothing can be received correctly. This depends on how strong both signals are at the receiver. When one of the signals is significantly stronger than the other, the stronger signal can be received normally. But when they are about equal, they will interfere with each other.

4.2 Jamming

In a jamming attack, the node that performs the jamming will try to prevent, or interfere with, the reception of signals at nodes in the surrounding WSN. It will do this by sending out a continuous random signal on the frequency that is used by the WSN. Affected nodes will not be able to receive messages from other nodes and will therefore be completely isolated until the jamming stops.

To see the effect jamming has on communication between two MICA2s, the following test set-up is used. Two nodes will try to communicate with each other. One node will try to send 100 messages and the second node will try to receive them. The receiving node will indicate how many of the messages it has received and how many messages contain errors, through its direct wired connection with a PC. A third node will act as a jamming node which can start and stop sending

jamming signals when instructed to by the PC it is connected with. The nodes are placed as shown in Figure 4.1

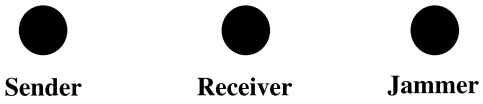
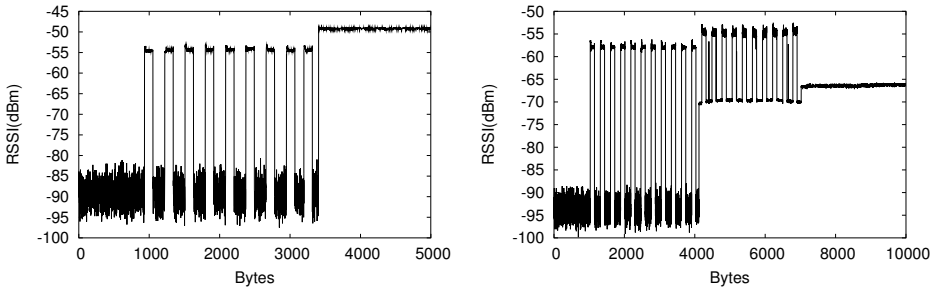


Figure 4.1. Placement of the nodes for the jamming experiment.

All messages are received correctly when running the test without the jamming signal. No reception problems were present before jamming started. When performing the test while a jamming signal is sent, none of the messages are received. The reason why no more messages are received can be seen in Figure 4.2(a). It shows the signals at the receiving side. The signal starts with background noise of around -90 dBm after which message transmission starts. This is when the RSSI rises to -55 dBm for a few bytes. After nine messages the jamming signal is started and all succeeding messages are overpowered by the signal.



(a) Jamming with a strong signal (-49 dBm)

(b) Jamming with a weak signal (-66 dBm)

Figure 4.2. Jamming through the transmission of packets.

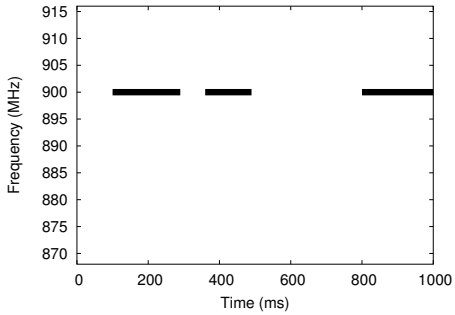
To show that jamming is only effective within a certain radio range, the node transmitting the jamming signal is moved away from the other nodes. The same test is performed and the resulting signal strengths can be seen in Figure 4.2(b). The jamming signal now no longer overpowers the transmitted messages but stays below the signal strength of the message. At this level all messages are being received correctly.

The advantage of a jamming attack is that no knowledge is needed of the WSN that is being attacked, except for the frequency the nodes are sending at. A disadvantage is the high power consumption of the continuous jamming signal, which causes the attacker to run out of energy rapidly.

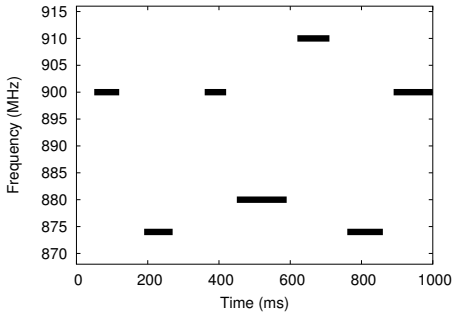
4.3 Countermeasures

When a jamming attack is preventing a node from receiving messages, it has a limited amount of options in defence. It can try to increase its send power in an attempt to get above the jamming signal at the expense of extra power consumption. But in return the jammer can do the same. Another option is changing the communication frequency. To be useful, all other nodes should do the same, but there is no way to tell the other nodes to change their frequency while a jamming attack is happening. This can be prevented with frequency hopping, where nodes change frequency in a predetermined sequence. If the jammer does not know this sequence, it can not follow to the other frequency. Frequency hopping is rarely seen in WSNs because of the extra complexity it involves. The MICA2 for example can only switch efficiently between two frequencies, and every extra frequency will require extra processing and calibration.

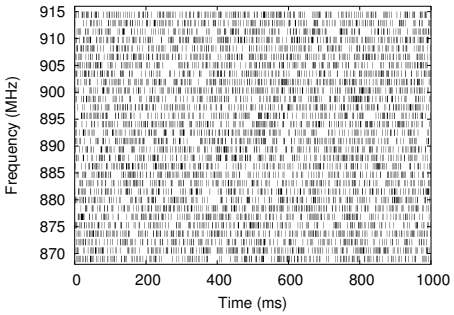
A radio communication technique that is virtually impossible to jam is Ultra Wide-band (UWB). UWB is based on the transmission of very short pulses in the order of nanoseconds, on a large part of a frequency band simultaneously. UWB is well suited for WSNs because of its low energy requirements and is therefore a worthwhile jamming countermeasure. Figure 4.3 shows an illustration of what data transmissions using normal frequency (Figure 4.3(a)), frequency hopping (Figure 4.3(b)), and Ultra Wide Band (Figure 4.3(c)) would look like.



(a) Single frequency.



(b) Frequency Hopping.



(c) Ultra Wide Band.

Figure 4.3. Data transmission using different send techniques.

Chapter 5

Link Layer

In this chapter, the attacks on the link layer are described in detail. In Section 5.1, the set-up and results of various collision tests are discussed. In Section 5.2, the possibilities concerning TinySec are discussed.

5.1 Collisions

In a collision attack, the attacker uses its radio to listen to the frequency a WSN is transmitting on. When it hears the start of a message it sends out its own signal interfering with the message. This is called a collision and causes the message to be received incorrectly at the receiver. In theory, causing a collision in only one byte is enough to create a CRC error and cripple the message. The advantage of a collision attack over a jamming attack is the power consumption. In a collision attack only a very short amount of time is spent on radio transmissions, which is more energy consuming than listening and therefore saves energy in comparison to a jamming attack. Another advantage is the difficulty of detecting a collision attack in comparison to a jamming attack. The only evidence of a collision attack is the reception of incorrect messages.

The tests to see how effective this attack can be is set up the same as the jamming attack described in Section 4.2. The node that was sending the jamming signal is now used to cause the collisions. The number of bytes that is sent during the collision ranges from one to six, while the byte in the message that triggers the collision ranges from byte number 13 to byte number 22. The results can be seen in Figure 5.1.

The figure shows both expected and unexpected results. The unexpected result is the poor effectiveness when sending only one byte. One would expect to achieve 100% message failure, but this is only achieved after sending four or more bytes. The reason for this is found when the RSSI traces of the attack are examined. When sending less than four bytes, a large percentage of the messages looks like Figure 5.2(a), while one would expect them to be like Figure 5.2(b). The spike of the collision is missing, which means that the node is not able to send out the first

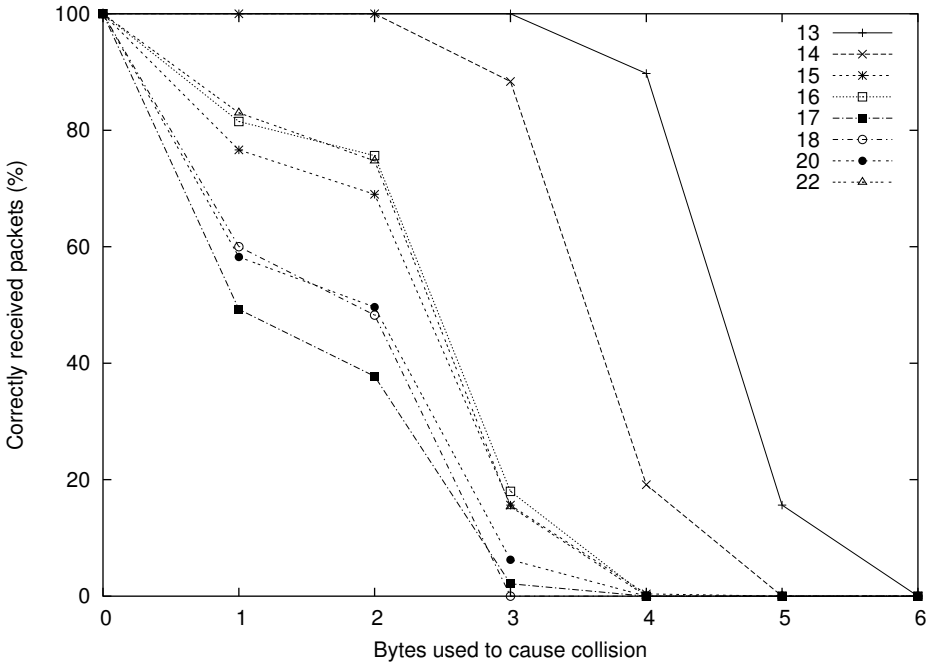


Figure 5.1. Reception of messages using a collision attack.

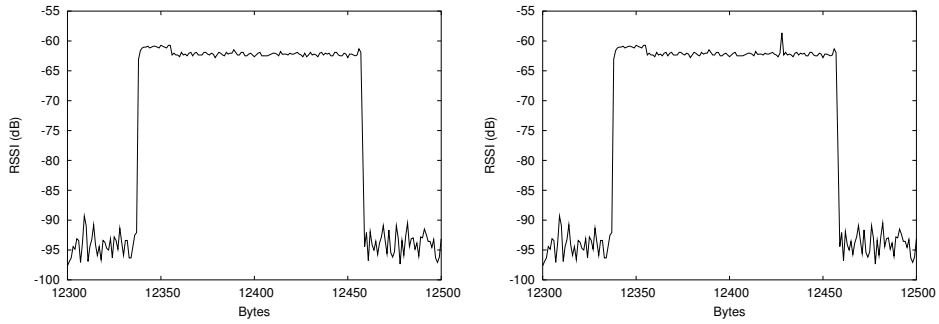
few bytes it is supposed to send. Only the fourth and later bytes are always being sent.

The correct reception of messages for bytes 13 and 14 is an expected result. Most protocols use a preamble. This is a number of bytes sent before the actual message to let the receiver know a message is about to be sent and to let the receiver adjust its radio to the incoming signal. It usually only needs two of these to be able to receive a message, but the messages in the test have 18 preamble bytes. Therefore, causing a collision in the preamble is not successful if more preamble bytes follow, which can be seen with bytes 13 and 14.

All countermeasures that can be used with a jamming attack can also be applied to collision attacks. The only difficulty is the detection of such an attack. Another precaution that can be taken is the use of an error correcting code, where errors in only a limited number of bytes can be recovered by using mathematical techniques [13]. If the attacker knows this it can respond by causing collisions with more bytes than can be recovered. When the structure of messages is known, a more effective collision can be generated, by attacking the length byte. This causes the protocol to receive the wrong number of bytes and no error correction can take place at all.

In short, to perform a successful collision attack, one should take notice of the following points:

- The signal must be strong enough to overpower the signal of the message;



(a) Correctly received message.

(b) Message with a collision in one of its bytes.

Figure 5.2. Different results when causing a collision with one byte.

- Bytes should not be sent in the preamble;
- Colliding with the length byte can prevent the recovery of wrong bytes with error correcting codes.

5.2 TinySec

A way to protect the messages sent at the network layer is to use encryption at the link layer. TinySec is the link layer encryption provided with TinyOS [3]. It can protect communication by encrypting the contents of messages, making it unreadable for anybody not in possession of the correct decryption key. It can also provide integrity by using a Message Authentication Code (MAC) that ensures messages are not modified during transit. When only using the MAC, the content of the complete messages can be seen, when also using encryption, only the message headers are visible. TinySec uses Skipjack [4] as the encryption algorithm, which uses a 80-bit key for encryption. While there are attacks for parts of the algorithm [5], no attack for all parts of Skipjack exists. It is not feasible to find the key within a reasonable amount of time with a supercomputer let alone with a resource constrained node.

Chapter 6

Network Layer

In this chapter, the attacks on the network layer are described in detail. In Section 6.1, routing technologies and exploits for them are explained. In the succeeding sections, the applicability of different exploits on existing routing protocols is examined.

6.1 Routing

A routing protocol in a computer network tries to find paths from sources to destinations. This is done to allow the source and destination to share information using these paths. Most routing protocols strive to find the best path according to a metric, such as least number of hops, minimal energy or shortest distance. The paths in WSNs are usually set up towards a base station and therefore form a spanning tree.

There are two main routing technologies: link state routing and distance vector routing. In link state routing, every node has a map of all nodes in the network and their connectivity with other nodes. Every node then independently calculates its best next node to send to for every destination. This information is stored in its routing table. If changes to the map occur, they are shared between nodes. A disadvantage of using link state routing in WSNs is that it requires more storage and computations than distance vector routing. Distance vector routing nodes share their routing table with their neighbours and use the routing tables from their neighbours to update their own. If a neighbour's routing table shows a shorter route to a destination, a node will update its own routing table accordingly. Distance vector routing protocols are simple and efficient, especially in small networks, and are therefore more suited for WSNs.

There are multiple attacks that specifically target routing protocols. The most common are selective forwarding, sinkhole, HELLO flood, routing cycles, and misdirection attacks [10]. A short explanation of each attack is given next.

- **Selective forwarding**
  In a selective forwarding attack, a node will forward most messages and

selectively drop, which means throw away, some of them. The attack is sometimes also referred to as neglect, because a node neglects to forward a message from a selected node or nodes. A selective forwarding attack can be combined with other attacks that try to draw in traffic, such as a sinkhole attack.

- **Sinkhole**
  In a sinkhole attack, a node tries to draw in all traffic or as much as possible from the surrounding network. It usually does this by making the node look very attractive to the surrounding nodes with respect to the routing metric or make it look like a base station. The surrounding nodes will then try to route their traffic through the sinkhole node, giving that node a lot of influence in that part of the WSN.

- **HELLO flood**
  In a HELLO flood, the attacker uses a powerful radio transmitter to send messages to the whole network. The messages are supposed to convince all nodes to choose the attacker as their parent and have them send messages to the attacker which is probably out of range.

- **Routing cycles**
  When a routing cycle occurs, the path from a destination to a source has a cycle in it. This means that a message will go around in circles, possibly forever.

- **Sybil attack**
  In a Sybil attack a node assumes multiple identities. It is usually done to fill a neighbouring node's memory with useless data from non existing neighbours. If a node has a limit on the number of nodes it keeps data of, it can be used to overwrite useful data.

6.2 Multihop

Multihop is one of the first and most simple routing algorithms for TinyOS. It is included with TinyOS 1.1 and later. It is a shortest-path-first algorithm, which means it gives priority to routes that are the least number of hops to its destination, which, in Multihop, can only be the base station. It also uses two way link estimation, which is an estimation based on both the quality of the communication coming from the node and the communication going towards it. These are called the receive quality and the send quality. The receive quality is calculated using an exponentially weighted moving average [20] using an α of 0.25. It uses the fraction of correctly received messages expressed in a range from 0 to 255. The receive quality values are sent to the neighbours who use them as their own send quality. These quality estimates are only used as a tiebreaker for routes with the same length. This effectively means that for example a route directly to a base

station with very high message loss will be chosen in favour of a two hop route with no message loss at all, which is not an unlikely case in a real WSN. Figure 6.1 shows an example of how routes are set up in a test WSN using Multihop.

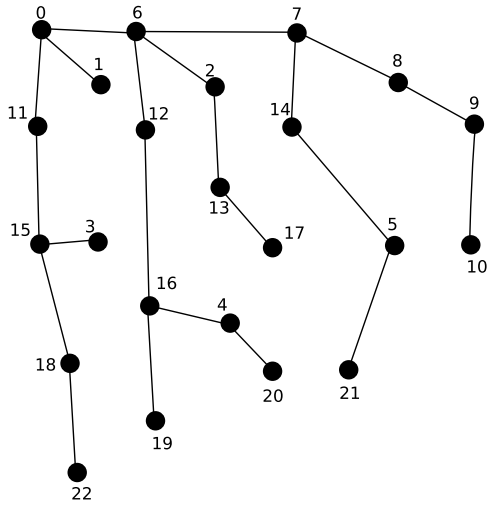


Figure 6.1. Routes in a WSN using Multihop.

6.2.1 Selective Forwarding

One of the easiest ways of trying to influence the communication in a Multihop network is selective forwarding. Even without knowing anything about the contents of the messages, this attack can be effective. When a node starts forwarding messages without dropping any, it will actually act as a repeater. This will result in neighbours of the repeater to conclude they are within radio range of each other, and thus only one hop away. Routes going through the repeater will therefore look shorter than they really are, which is an advantage in shortest-path-first algorithms. A simplification of this can be seen in Figure 6.2. Node A receives messages from both B and D, but will choose node B as its parent because it appears to have a shorter route.

Simulations showed that a node that acts as a repeater can draw in more routes than when it would be participating normally. For example, the forwarder in Figure 6.2 will always be chosen when forwarding and only half of the time when participating normally. Repeating all messages that are received is not an effective attack on a network. On the contrary, as long as a node keeps forwarding all messages it receives, it is only beneficial to the network, because its resources are used for sending useful messages for the network. Therefore, at some point in time, messages can be dropped by the forwarder, hence the name selective forwarding. When deciding which message to drop, different approaches can be taken, depending on the amount of knowledge of the message contents and the used routing algorithm.

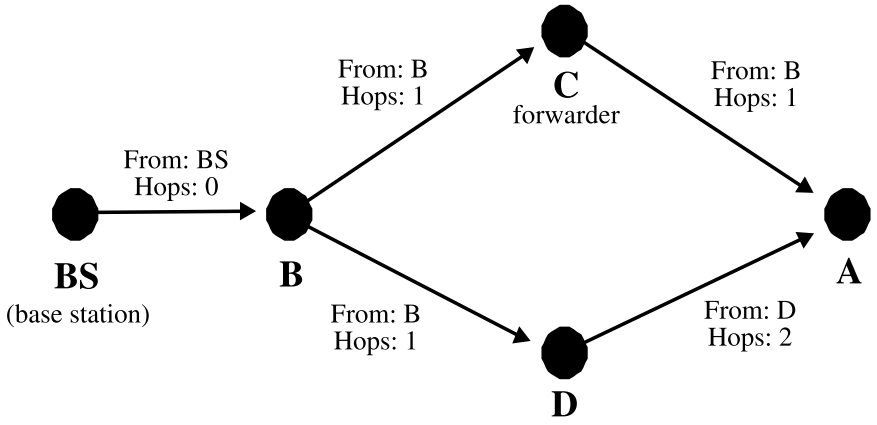


Figure 6.2. Routing updates going towards a node in a WSN.

The effectiveness of an attack is dependent on two factors. The first is the placement of the malicious node. The closer it is to the base station, the more traffic will pass through it. The second is the fraction of messages that are dropped. When too many are being dropped, the link quality will keep degrading and eventually drop below a certain threshold after which Multihop will ignore the node in subsequent parent selections. To see where this threshold lies, the WSN in Figure 6.2 is used with different dropped message percentages. Figure 6.3 shows how many messages are sent through the forwarder when all nodes have perfect reception. In this particular case, the selective forwarder node can drop around 35% of the messages from node A without node A changing its parent.

The amount of energy needed for the attack depends on the same factors of placement and percentage of messages dropped. The more traffic that passes through the forwarder, the more energy is needed, but it will never be significantly more than other nodes in the network. And every message that is dropped will save the energy the attacker would otherwise spend on forwarding that message.

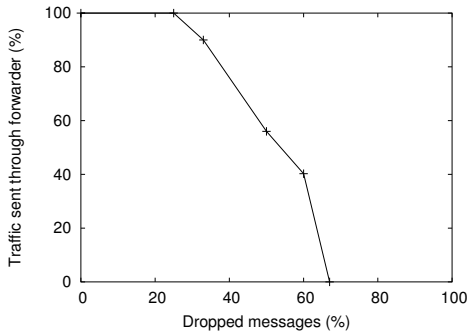


Figure 6.3. Percentage sent through forwarder.

6.2.2 Sinkhole

With more knowledge about the routing algorithm, it is more effective to use this knowledge to take part in the routing mechanism than to use it only for selective forwarding. When taking part in routing, a sinkhole can be created. The easiest way of creating a sinkhole is to have a malicious node pretend it is a base station. This can cause a big part of the network to start sending their traffic towards that node. How many nodes are affected depends on the part of the network the malicious node is located in. This is because nodes closer to the real base station will not send traffic towards the malicious node because it is further away than the real base station. Figure 6.4 shows the routing paths in a WSN during a sinkhole attack by node 3.

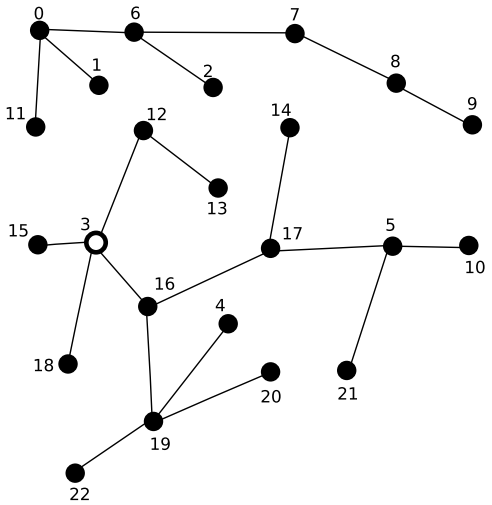
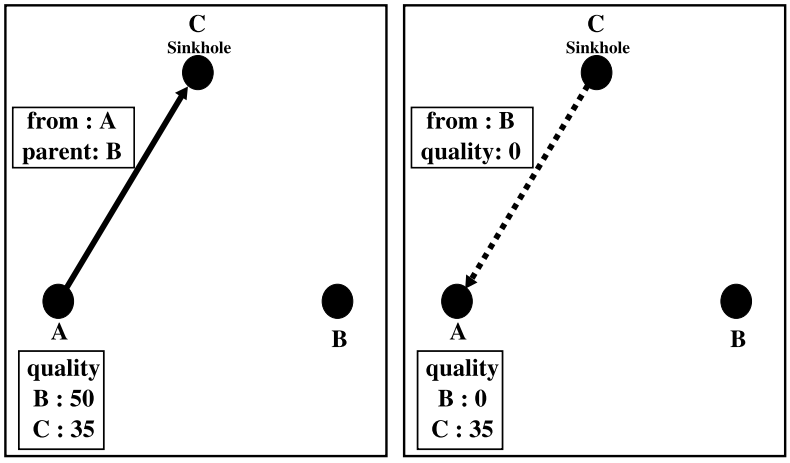


Figure 6.4. A WSN while node 3 is performing a sinkhole attack.

Multihop nodes periodically send out route updates with information for their neighbours. These messages contain the node's identifier (ID) and the hop count to the base station, and also for every node in its neighbour table the receive quality value indicating the quality of the reception from this node. When selecting a new parent, all nodes that have a link receive or send quality below 25 will be skipped from selection. When a node has a hop count of zero, which is the base station, its receive quality is ignored.

The route updates can be used against the network to get an even larger influence on the network and thus a bigger sinkhole. To achieve this, the node creating the sinkhole will listen to messages from nodes that indicate they have someone else than the sinkhole node as their parent. The sinkhole will now use the IDs from the sender, say node A, and its parent, node B, and forge a route update that looks like it is coming from node B and send it to A. In the route update, the send quality is set to a value below 25. Upon reception, node A will update the entry for node B in its neighbour table with the new values. When it tries to choose a

new parent it will now skip node B and hopefully pick the sinkhole node as its new parent. Experiments have shown that the sinkhole is able to make nodes switch to the sinkhole when performing this attack. The attack is illustrated in Figure 6.5. If between the sending of the fake message and the selection of the new parent a real route update from node B is received, the attack will not work because the send quality will be set to its correct value. If node B itself is within radio range of the sinkhole node, the sinkhole node can wait with its message until after it hears node B send its route update.



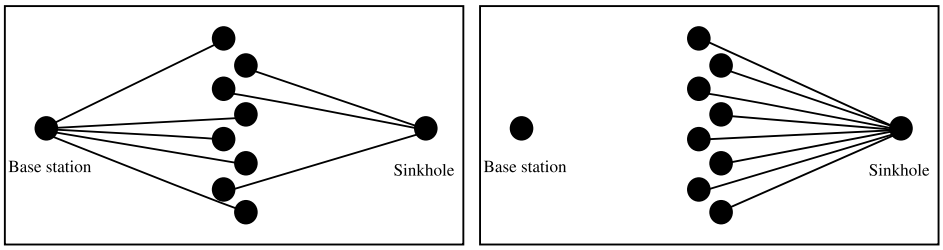
(a) Node A broadcasting its parent.

(b) The sinkhole sends a message to A with node B as source telling node B has a bad link quality.

Figure 6.5. WSN before and after an extended sinkhole attack.

While the previous approach can be used on other routing protocols as well, Multihop has a more specific mechanism that can also be exploited. Every node puts a sequence number in all messages it sends, and receiving nodes compare this sequence number to the previously received sequence number from that node. It will use this to see how many messages it has missed and adapt its receive quality accordingly. If the sequence number has not gone forward but backwards by a value larger than 20, the receiving node will conclude something has gone wrong with the node and resets all the values in the corresponding neighbour table entry, including its send and receive quality. When using the same setup as in Figure 6.5, the sinkhole node can now send two consecutive messages that look like they are coming from node B. The second message must have a sequence number more than 20 less than the sequence number in the first message. Upon reception of the second message, node A will reset the receive and send quality to zero. A real route update from node B will now change the send quality back to its correct value

but the receive quality will not reach a value above 25 because of one message. Therefore node B will still be excluded from the next parent selection. Simulation results in Figure 6.6 show a test WSN before and after such a sinkhole attack on the base station (node B).



(a) Status of a WSN during a normal sinkhole attack.

(b) Status after the sinkhole makes the base station look bad.

Figure 6.6. WSN before and after an extended sinkhole attack.

When a sinkhole attack is mentioned in the literature, it is often combined with a selective forwarding attack. This would create a sinkhole node that has control over virtually all traffic in the WSN. However, the nature of a sinkhole attack makes this almost impossible. To have control over the traffic towards the base station, the sinkhole node first has to make all nodes route its messages through the sinkhole. But when the sinkhole node wants to send a message along to its intended destination it runs into problems. It is incapable of accomplishing this, because every node will send all the messages it receives back to the sinkhole. Therefore these combined sinkhole/selective forwarding attacks are actually impossible.

6.2.3 HELLO Flood

The preference for the shortest route can usually be exploited by a HELLO flood. In the case of Multihop, this means broadcasting a message with a long-range radio-antenna to all nodes in the network, stating the node performing the HELLO flood is the base station. The receiving nodes should then conclude that the route through the node sending the HELLO flood is the shortest. They will try to send all their succeeding messages through this node, which most probably is not even within radio range. In the ideal case, all nodes in the network will keep sending their messages into oblivion and waste their energy. The problem with this attack is that the real base station will be broadcasting the same type of messages, with the difference being that fewer nodes will probably hear these. This means that at least some nodes will have a choice between the real and the fake base station.

The effectiveness of a HELLO flood attack will be tested in the following set-up. A HELLO flood node is added to a normal network topology. The HELLO flood node will act the same as a normal base station and therefore only know the

IDs of its neighbours, but it can reach all nodes in the network, while only nodes that have the HELLO flood node within their radio range can send messages back.

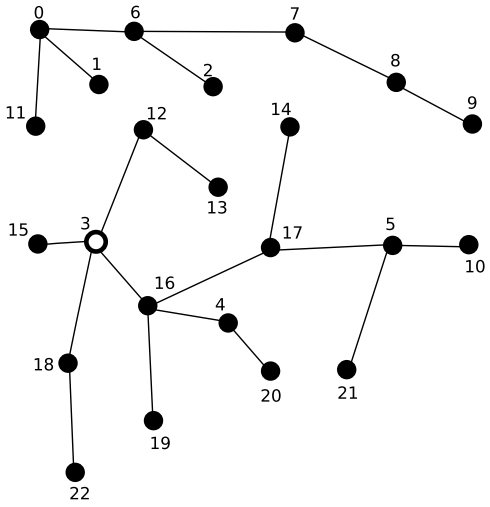


Figure 6.7. HELLO flood by node 3 in a WSN topology.

Simulation results in Figure 6.7 show that not all nodes will choose the attacking node as its parent. Some of these nodes are next to the base station and will choose the base station as their parent as expected, but other nodes are multiple hops away. The cause of this is the content of the route update. A node will only consider a node as a possible parent if it has both a receive and send quality above 25. The send quality value needs to be included in a route update from the parent. Therefore, an attacking node needs to include a, possibly fake, send quality for all nodes in the network. The attacking node can not know all node IDs in the WSN because not all nodes are within radio range. If it wants to reach the complete network it must guess the node IDs. If the HELLO flood node has unlimited power, it can simply include all possible node IDs in multiple messages of the maximum allowed length. This becomes almost impossible when there are a lot of possible IDs. The attacking node also has to keep sending its messages because when it stops the nodes will start gradually lowering the receive and send quality value until it reaches zero.

It is clear that a HELLO Flood in Multihop is hard, if not impossible, to perform. It will require a lot of energy and messages and will never be able to affect all nodes with absolute certainty.

6.2.4 Routing Cycles

Multihop contains a check for routing cycles. Every message in Multihop contains the number of hops it still has to travel to the base station. Every node that forwards the message decreases this number by one. Because a node knows the number of hops from itself to a destination it will compare this with the number of hops that

is in the message. If the number of hops in the message is smaller this may indicate a routing cycle or another error and the message is discarded. Therefore, there is no possibility to create a cycle in Multihop.

6.2.5 Sybil Attack

Multihop keeps data of its neighbours in a neighbour table. The table has a maximum size of sixteen and therefore it is impossible to completely fill a node's memory. However, tests show it is possible to fill this neighbour table with useless data. When the table is full and a message from a node that is not in the table is received the node with the lowest send quality is replaced with the new node. If a Sybil attack node assumes the identity of sixteen nodes it can remove all real neighbours from the neighbour tables of all nodes within its radio range. It can even remove the base station if the fake node's send quality is higher than the one from the base station. The down side of this attack is the large number of messages that will need to be sent.

6.2.6 Alternative Attack

An attack such as the HELLO flood is meant to completely disable the WSN and prevent it from performing its tasks. If one has access to a long-range radio-antenna like the one used in a HELLO flood, a Multihop WSN can be more effectively disrupted with another attack. Instead of convincing all nodes to send to a node they can not reach, one only tries to influence the nodes that have the base station as a parent. The whole network relies on these nodes to forward their data, and therefore, disabling these nodes affects the whole network. To achieve this, the nodes need to skip the base station in their next parent selection round. The same technique used in the sinkhole attack can be used for this, in which two consecutive messages are sent, where the second message has a sequence number twenty less than the first. The nodes will now skip the base station in their parent selection and no data will ever reach the base station. Figure 6.8 shows the status of a WSN topology during an experiment where this attack is being performed.

6.3 Mintroute

Mintroute [22] is an adaptation of Multihop. The only real difference between Multihop and Mintroute is the selection of the parent. In Multihop, the least number of hops is used whereas in Mintroute, the link quality (as described in Section 6.2) with the surrounding neighbours is used together with a cumulated route quality to the base station, and the hop count included in the route updates is completely ignored. This can lead to undesirable results, when a node has more than one neighbour with the same highest link quality. Mintroute will then arbitrarily choose one of them as its new parent, which could be in exactly the opposite direction of where the base station is. The natural occurrence of these suboptimal routes

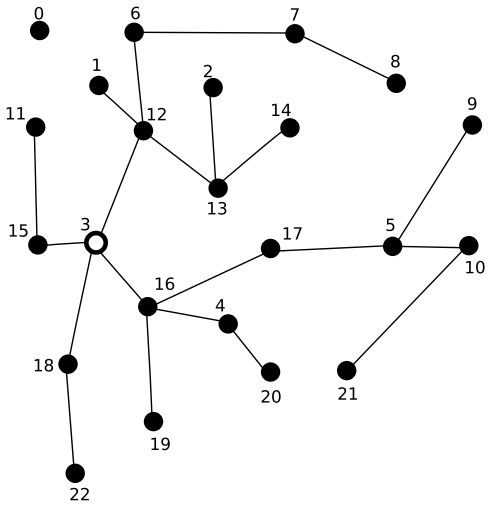


Figure 6.8. A WSN while node 3 is performing an alternative attack.

should be taken into account when performing simulations. Figure 6.9 shows an example of how routes are set up in a test WSN using Mintroute. The route from node 8 to 7 via 14 is one of the suboptimal routes.

Because of the similarities between Multihop and Mintroute, the attacks and their simulation results will be discussed more briefly and only important differences will be reviewed more thoroughly.

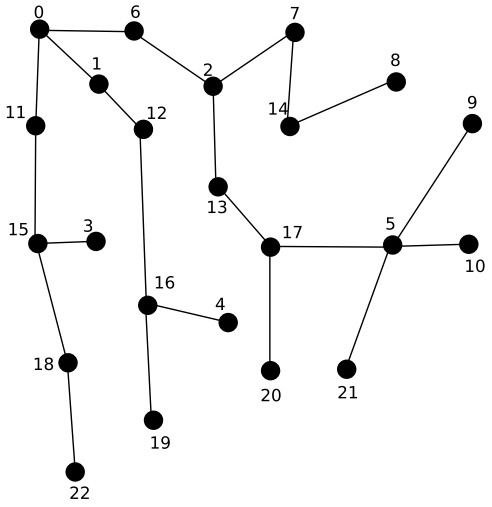


Figure 6.9. Routes in a WSN using Mintroute.

6.3.1 Selective Forwarding

A selective forwarding attack in Mintroute can be set up the same as described in Section 6.2.1. The attack is less effective with Mintroute. The reason for this is the different parent selection, based on link quality instead of hop count in Multihop. When the attacking node drops messages, the link quality will degrade and surrounding nodes will choose other nodes with a higher link quality. While nodes in Multihop can use a shorter route through the forwarder node and prefer this shorter route, this advantage does not apply for Mintroute nodes. In Mintroute the extra node a message has to go through only reduces the already present link quality, which makes the selective forwarder look less attractive. This is shown in Figure 6.10.

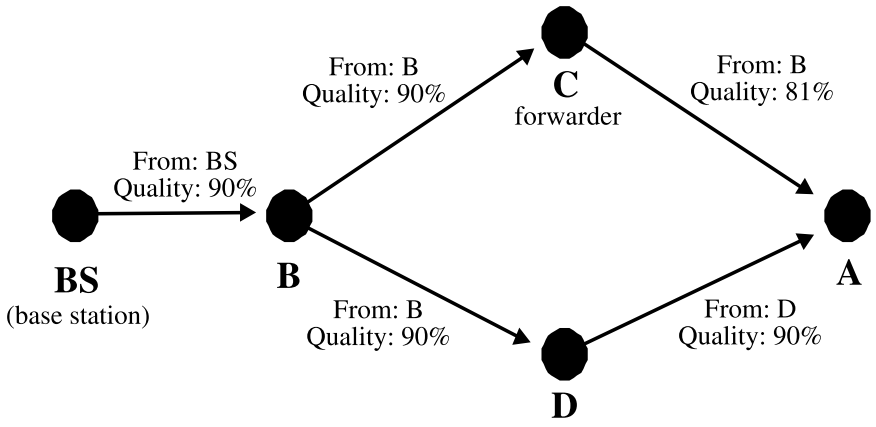


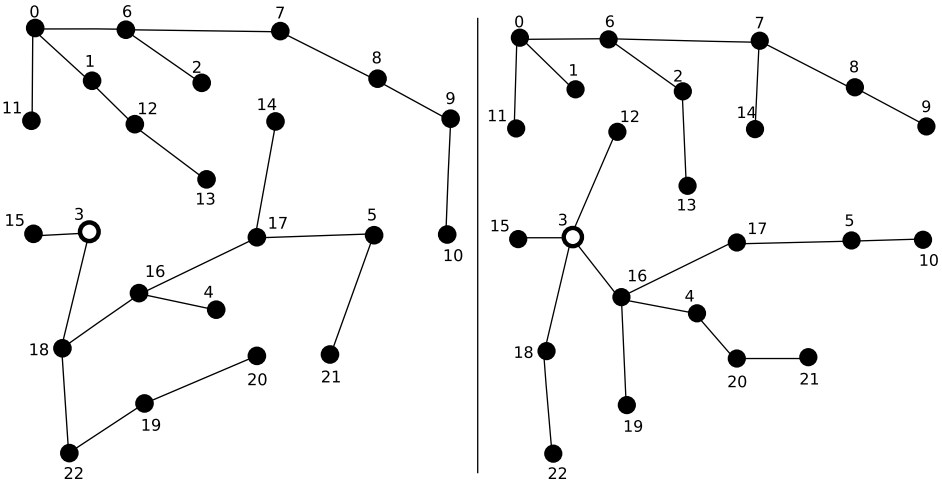
Figure 6.10. Link quality seen by nodes in a WSN.

The figure has the same set-up as Figure 6.2 with the hop count replaced by the link quality as it is perceived by the nodes. Node A will have to choose between node B with a link quality of 81% or node D with a quality of 90%. This illustrates why the quality metric in Mintroute is harder to exploit using selective forwarding than the hop count metric.

6.3.2 Sinkhole

Creating sinkholes in Mintroute is less straightforward than in Multihop. When a node has the base station as one of its neighbours, the node will not automatically choose it as its parent. Instead, it will choose the neighbour with the best link quality. This means pretending to be a base station is not going to attract extra traffic. To be chosen, a node must have both a good send and receive quality. To get a high send quality, the high value must be included in a route update sent by the sinkhole node. To get a high receive value, the sinkhole will have to keep

sending messages to prevent the decaying of the receive value by the node. The number of messages that are lost also lowers the receive quality, but besides using its maximum sending power, the sinkhole has little influence on this. Simulation results from such an attack can be seen in Figure 6.11(a).



(a) Status of a WSN during a normal sinkhole attack.

(b) Status after the sinkhole makes other nodes look bad.

Figure 6.11. WSN during two different sinkhole attacks using Mintroute.

The sequence number approach used with Multihop described in Section 6.2.2, can be applied to a Mintroute sinkhole as well, which can be seen in Figure 6.11(b). This works with Mintroute because it will reset the send and receive quality from other nodes. Thus, instead of increasing its own send and receive quality the sinkhole lowers those of its competitors.

6.3.3 Other Attacks

The HELLO Flood, routing cycle, Sybil attack, and alternative attack as presented in Sections 6.2.3, 6.2.4, 6.2.5, and 6.2.6, respectively can be applied unchanged to Mintroute. The only note is the fact that in a HELLO Flood attack, the attacker needs to keep its rate of messages being sent high to sustain its high receive quality, which is necessary for a successful attack.

6.4 DSDV

DSDV, or Destination-Sequenced Distance-Vector routing [17] is a routing algorithm that is part of the Heterogeneous Sensor Routing (HSN) stack that can be found in the contrib section of TinyOS. The DSDV routing layer protocol provides

many-to-one routing to one destination at a time and can be used with either a hop count metric or a quality metric.

In DSDV, the base station will send out an advertisement with its identifier, metric, and a sequence number periodically. All nodes receiving this message will forward it and use the identifier as their target destination for the routing algorithm. This will cause the network to set up a new spanning tree every time a route advertisement is sent.

6.4.1 Hop Count Metric

In the hop count metric in DSDV, the nodes use the route to the destination that has the least number of hops they know of. This is the same as Multihop but without the link quality as a tiebreaker. Instead it will only switch to another node if it has a lower hop count or if it detects it has missed more than three route updates for the current node.

Selective Forwarding

Selective forwarding will work as expected and even draw in more traffic just as in Multihop because of the shorter routes it will create. Because bad send and receive quality is not considered when choosing a parent, the forwarding node only has to make sure it forwards enough route updates to prevent switching to a new parent. It can drop any other message without the routing protocol detecting it.

Sinkhole

In a sinkhole attack in DSDV, one can take two approaches. The sinkhole node can pretend to be a base station and start sending out advertisements with its own identifier. Nodes in the network will change their destination node to the sinkhole's ID and try to find the shortest path towards it. All nodes in the network will be affected and start sending their messages towards the sinkhole but as soon as the base station sends another advertisement all nodes will change their destination back to the real base station. The sinkhole will have to resend its advertisement to make the nodes change destination again. This attack will affect all nodes in the network but this will not be permanently. This can be seen in Figure 6.12(a), where the percentage of the messages sent towards the sinkhole can be seen for all nodes in the network topology shown in Figure 3.3.

The other approach is pretending to be the base station or having the best route to the real base station. This will cause all nodes closer to the sinkhole than to the base station to switch their route to the fake base station instead. Because the nodes do not switch the destination id, they will not change back when an advertisement from the real base station is received. This approach will not affect all nodes, but the nodes that are affected will stay that way until the sinkhole stops sending advertisements. Figure 6.12(b) shows that the network is divided in nodes sending to the base station and nodes sending to the sinkhole.

DSDV uses sequence numbers to indicate the order of the messages that are sent. It uses the sequence numbers to skip old or duplicate messages and to detect missed messages. The detection of old messages can be exploited to get a better sinkhole. When a message is received that has a sequence number that is not more than 64 less than the sequence number last received, it is regarded as an old message and ignored. The sinkhole can now copy a new route advertisement from the real base station and increase the sequence number by more than three. On arrival at nodes in the network it will first cause a new round of parent selection, because the nodes will think they have missed three route updates from their parent. In this parent selection, the route to the sinkhole will be chosen instead of the route towards the base station. When a route advertisement from the base station arrives, it will have a sequence number that is lower than the one last received and the messages will be ignored even though it could have a lower hop count. If the sinkhole keeps increasing the sequence number, and keeps it above the sequence number from the base station, it will attract all network traffic. The sequence number increase can be done with steps of 64 without the attack failing. This means for every 64 route advertisements the base station sends, the sinkhole has to send only one and still keep control over the network. Figure 6.12(c) shows how all nodes are sending their messages to the base station during this attack.

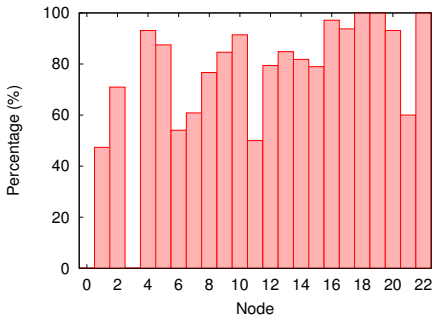
As discussed in Section 6.2.2 it is not possible to get the messages from the sinkhole to the base station during the attack and manipulate the communication instead of disabling it.

HELLO Flood

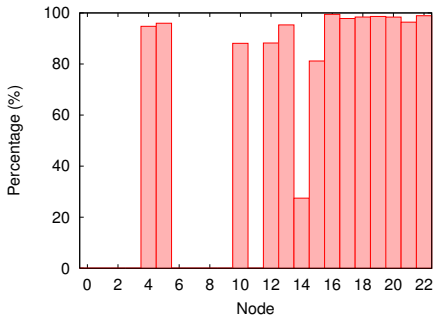
When performing a HELLO flood, the same options as in the sinkhole attack are available. One can either send messages to all nodes from a base station with a new ID or pretend to be the real base station. The option of pretending to be the real base station is the most effective attack. Nodes will still choose the real or fake base station that is the closest, but this will only be the nodes that have the real base station as its neighbour. After the first route advertisement from the fake base station, all nodes in the network will start sending their messages directly to the fake base station most nodes can not reach, which puts the network in a disabled state for as long as the fake base station keeps broadcasting route advertisements. When using the other option of a second base station with a new identifier, the nodes in the WSN will only be affected until the other base station sends out a new route advertisement. Simulations confirm that this behaviour takes place.

Routing Cycle

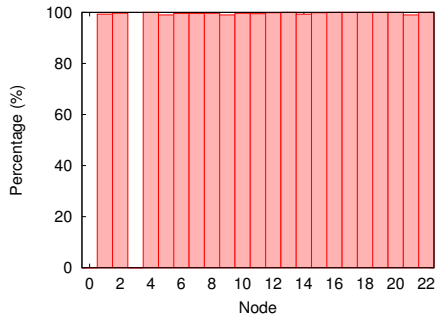
DSDV with the hop count metric has no explicit check for routing cycles. Therefore, an attempt will be made to create a routing cycle. The test set-up is shown in Figure 6.13(a). It shows part of a WSN network where all messages towards the base station are going to the left through node A. The figure shows how many hops



(a) Sinkhole 1: Use a second base station.



(b) Sinkhole 2: Pretend to have the best link with the base station.

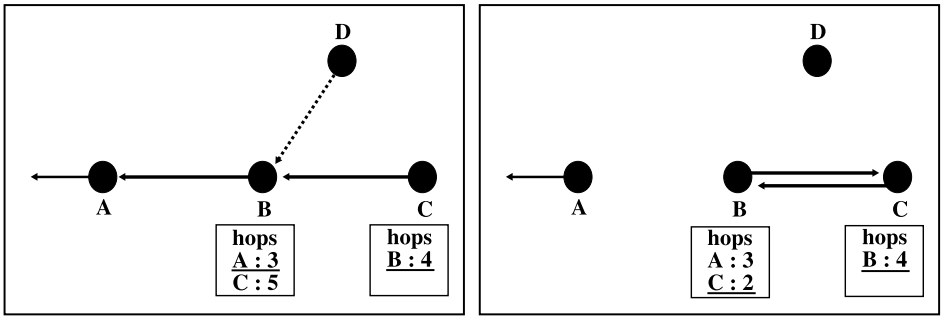


(c) Sinkhole 3: Desynchronise the base station.

Figure 6.12. Percentage of messages from each node that is sent to the sinkhole (node 3).

to the base station node B and C think their neighbours have. The node they have chosen as parent is underlined.

In the attack, node D will copy a route advertisement coming from node C and lower the hop count to a value lower than the hop count to node A and broadcast it as if it is coming from node C. Figure 6.13(b) shows the status after B has received the message. Node B thinks it has found a better node to which to send and will send all its messages towards node C. Node C still thinks node B is the best node to which to send and will forward all incoming messages to it. This will create a cycle in the network and node B and C will keep sending the same message to each other until they run out of power. The attacking node only has to send one message to start the cycle and no further messages are needed to keep it running. If the attacking node is mobile, it could theoretically make all n nodes in a WSN waste their energy rapidly with only $n/2$ messages.



(a) The status of the unaffected WSN.

(b) The status after B has received a fake message claiming C is only two hops away from the base station.

Figure 6.13. The results when creating a cycle in part of a WSN.

6.4.2 Quality Metric

When the quality metric is used in DSDV, a node will include the value it has of the quality of the route to the base station in a route advertisement. All nodes keep track of the link quality with all their neighbours, which is defined by the number of messages from the last 32 that were not received. When a node receives a route advertisement from a node, it will take the link quality of that neighbour and add it up with the quality included in the message. The nodes will choose the node with the lowest value as their parent. With equal values they stay with their old parent.

Selective Forwarding

Selective forwarding with a quality metric instead of a hop count metric is more difficult to perform as was already discussed in Section 6.3.1. An extra hop will introduce extra lost messages and therefore a less desirable link quality. Dropping messages will make the link quality through the forwarder look even worse, which will cause nodes to route around the selective forwarder.

Sinkhole

Using the quality metric instead of the hop count has no effect on the sinkhole attacks effectiveness. The only difference will be the boundary in the network where nodes will choose the real base station instead of the sinkhole. With a hop count metric this will be where nodes have the same number of hops towards both the base station and the sinkhole. The boundary when using the quality metric will be where the quality to both nodes will be about equal. This boundary is a lot harder to determine and is not necessarily the same as the hop count boundary. This boundary determines the number of nodes that are affected by the attack. Figure

6.14 shows which part of the test WSN sends its messages to the sinkhole with the quality metric.

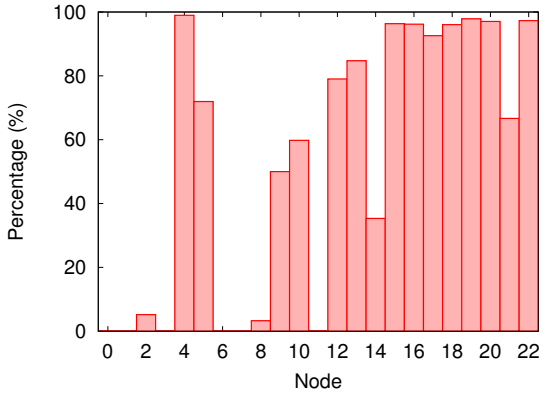


Figure 6.14. The percentage of messages sent to the sinkhole.

HELLO Flood and Routing Cycle

The HELLO flood and routing cycle attacks will still work the same with a quality metric as long as a value that corresponds to the best available quality metric instead of the hop count metric is included in the fake messages. In the DSDV quality metric, zero corresponds to the best quality metric.

6.5 TinyAODV

TinyAODV is a simplified version of the Ad-Hoc On Demand Vector routing algorithm [18]. It is an on demand algorithm which means it will try to construct a route between a source and a destination when it is needed. When a node wants to create a route to another node it will broadcast a route request that will travel across the network. When it reaches the destination node a route reply is generated and sent along the reverse route the route request travelled. All nodes along the reverse route will add the route to their route table, which has 7 entries. The routes never expire but when the route table is full, old entries will be replaced. The routes are set up using a simple hop count metric.

TinyAODV is also part of the HSN stack from the contrib section of TinyOS. The code of the implementation used for testing had to be adjusted to make it functional. When choosing the route back to the source, the algorithm would use the neighbour with the highest hop count instead of the lowest and therefore, create suboptimal routes. This is corrected in the code used for testing.

6.5.1 Selective Forwarding

As has been discussed in previous selective forwarding attacks using a hop count metric such as in Section 6.4.1, a selective forwarding attack in TinyAODV will shorten routes and therefore potentially attract more traffic. The difference is the fact that traffic can have its source and destination anywhere in the network. There can be, for example, two base stations instead of one. This means that if the forwarding node is close to one of the base stations, it will receive almost nothing of the traffic towards the other base station.

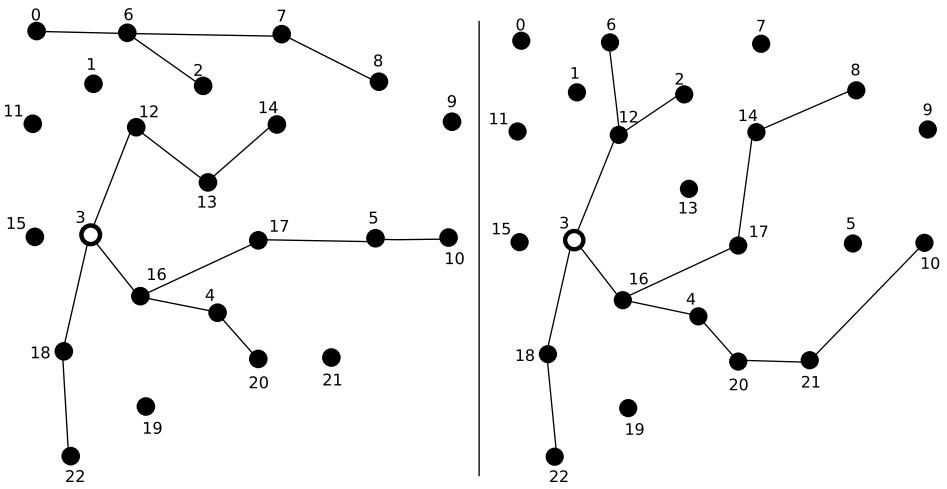
6.5.2 Sinkhole

Because in TinyAODV there is no single destination for messages, such as a base station, a sinkhole can not imitate a node and attract traffic towards it. To get as much influence on routes as possible, a sinkhole will have to take action every time a route is being created. When a sinkhole receives a route request, it will respond by directly returning a route reply, indicating it has found the destination with the lowest possible hop count, which is zero in TinyAODV. All nodes along the route back to the source of the route request will store the route towards the sinkhole in the routing table. If the real destination is a larger number of hops away from the source than the sinkhole, the route to the sinkhole will be chosen. When the real destination is closer, the route to the sinkhole will be ignored. The effect of a sinkhole in a WSN is shown in Figure 6.15(a). In this instance all nodes with an even number tried to set up a route to node zero. Only the even numbers were used to prevent strange result when the routing tables would be to small for all nodes. The lines indicate the routes that were actually created.

One of the differences between normal AODV and TinyAODV is that routes never expire in TinyAODV. This means nodes will never have to resend a route request because a route has not been used for a long time. In TinyAODV, route request messages still contain a sequence number to distinguish between different attempts to set up a route to the same destination. But these sequence numbers are not used because the route never expires, and therefore the sequence number is always zero in TinyAODV. A sinkhole can exploit these sequence numbers by sending out route replies with this sequence number set to one. All route requests and replies with a lower sequence number are now ignored by all the nodes in the network. Because TinyAODV only uses zero as a sequence number, the whole WSN will only listen to the routing messages of the sinkhole. Figure 6.15(b) shows the routes that will be created when the sinkhole node sends replies with a sequence number of one instead of zero.

6.5.3 HELLO Flood

If a sinkhole node can reach all nodes in the network, it can send a route reply directly to a node as soon as an attempt to create a route by this node is detected. Because, to the receiving node, the sinkhole will look like only one hop away, it



(a) A normal Sinkhole attack performed by node 3.

(b) A sinkhole attack with a different sequence number.

Figure 6.15. The result of sinkhole attacks in TinyAODV when nodes with even numbers try to reach node zero.

will choose the sinkhole as the node to send to. Unless the real destination is also only one hop away. Simulations confirm that almost the complete WSN can be put into a disabled state this way.

6.5.4 Routing Cycle

In [18], the authors present a proof why the AODV algorithm is loop free. While the proof is indeed correct, there is no guaranty that loops can not be deliberately created. Nodes in TinyAODV broadcast their route information without an indication from who they received it. This makes it easy to send messages that look like they originate from another node. Because of this, routing cycles can actually be created without problems.

To create a routing cycle between two neighbour nodes B and C, both a route request and a route reply need to be created by an attacking node D, which is shown in Figure 6.16.

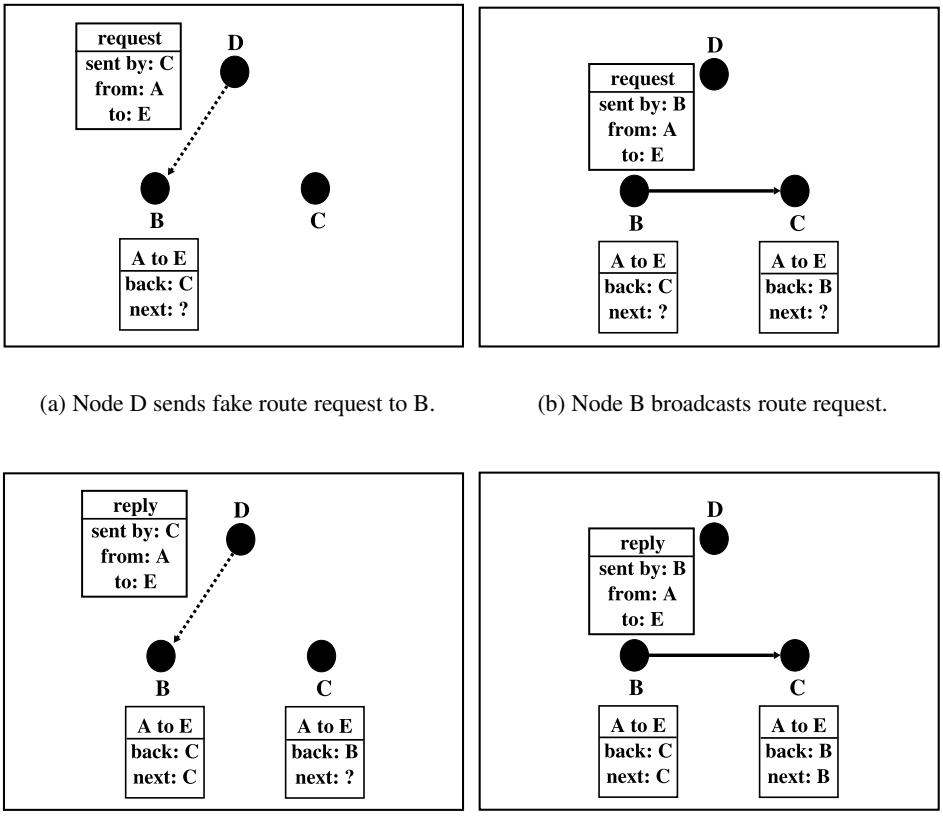
- First, a fake route request is sent from D to B that looks like it comes from C and that indicates a route needs to be set up between two arbitrary nodes A and E (see Figure 6.16(a)). Node B will remember node C as its backwards hop in the route from A to E and rebroadcast the request.

- Node C will receive this broadcast and remember node B as its backwards hop in the route from A to E (see Figure 6.16(b)).

- Node D will now send a route reply to B, that indicates that node C has found a route to E and that the route needs to be stored. Node B stores the node that has sent the reply in its routing table as its next hop towards node E (see Figure 6.16(c)). In this case this will be node C.
- Node B will then resend the reply to the node that it remembered as its backwards hop, which is also C. Upon receiving the reply from B, node C will do exactly the same as node B and store B as its next hop towards E (see Figure 6.16(d)).

All node D has to do now to start the cycle is send a normal message to B that looks like it is destined for E. Node B and C will now forward the message between them forever or until their power supply is depleted.

This means a node only needs 3 messages to disable two nodes and have at least one of them use up all its energy.



(a) Node D sends fake route request to B.

(b) Node B broadcasts route request.

(c) Node D sends fake route reply to B.

(d) Node B resends route reply towards C.

Figure 6.16. Node D creates a routing cycle between node B and C.

Chapter 7

Discussion

Attacks and countermeasures in wired and wireless networks can be seen as a cat and mouse game. Every attack asks for a new or different defence and every defence for a new or different attack.

The easiest attack on a network would be the jamming attack from Chapter 4. This could be performed with a very basic radio tuned to one frequency and a large power supply without the need for a processor, memory or other complicated equipment. But as soon as the nodes start switching frequency as a countermeasure, the attack becomes useless and more advanced knowledge is needed to keep attacking. Using collisions (Chapter 5) instead of jamming could prevent the nodes from switching frequency because this attack is much harder to detect by the nodes. This is because nodes can listen to their radio to detect jamming, but they can not listen while they are sending a message, so collisions can not be observed.

When the attacker has more knowledge about the network, such as the routing algorithm that is being used, it can try to reach nodes beyond its radio range, which has been demonstrated in Chapter 6. It can potentially try to disable a large network with nodes far beyond the range of a jamming or collision attack, while using less energy. A possible countermeasure against these attacks would be the use of encryption algorithms, such as TinySec from Section 5.2. Unfortunately, this leads to a higher energy consumption by the nodes and it will not stop selective forwarding attacks because these do not change the encrypted messages and the header is not encrypted.

It is very hard, if not impossible, to defend a WSN against all attacks. But not every WSN needs to be protected from all attacks. If the data from the WSN is allowed to be seen by the attacker, data encryption is not needed. For example, temperature readings from the different rooms in a building are very useful to a climate control system, but to an attacker this data is of no use.

Knowing where the vulnerabilities of a routing algorithm are can also be very useful when deciding what security measures to take. From Chapter 6 it is clear that some of the attacks on routing algorithms can be avoided with only small changes. For example, Multihop and Mintroute can be disrupted by exploiting

the sequence numbers used. These attacks are the most effective. Changing the algorithms so this can no longer be done will prevent the attacks that completely disable the network from working. This will assure that the WSN will never be completely disabled by a simple attack.

Both DSDV and TinyAODV are vulnerable to routing cycle attacks. This is because they lack any checking mechanism for this. Routing cycles can be very disruptive but adapting the algorithm to prevent them is fairly easy and will make a network more secure.

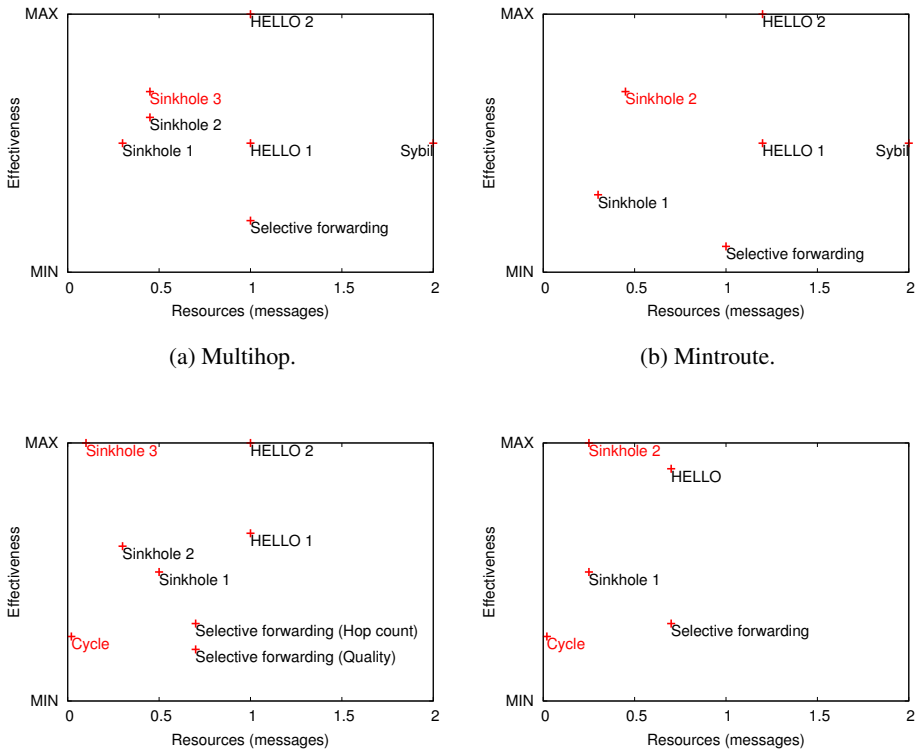
The most severe attacks on DSDV and TinyAODV are the attacks making use of the sequence numbers in these algorithms. The attacks can affect all the nodes in a WSN. The attack on TinyAODV actually exploits a bug in the code and can therefore be easily avoided when fixed. The attack on DSDV uses a mechanism that is used to skip old messages, and securing the algorithm against this will be harder but still worthwhile because it is the most disruptive and efficient attack on DSDV.

Determining how effective an attack really is and how it relates in effectiveness to other attacks is very hard. The effectiveness depends on a lot of different factors, such as the used topology, the location of the attacker(s) in the network and the rate at which messages are sent, but also for example if a message is sent just before or after the base station has sent a message such as a route advertisement.

Figure 7.1 shows how the attacks on the different routing algorithms in TinyOS relate to each other. It shows both the effectiveness and required resources for every attack. The effectiveness is an indication of how many of the nodes in a network can be affected by the attack. The effectiveness is somewhere between MIN and MAX because no exact values can be given. The exact value depends on the different factors mentioned above and would therefore only hold for a specific set-up. The same applies to the resources that indicate the number of messages that have to be sent by an attacker in comparison to other nodes in the network. While these values are not exact, they do indicate that if an attacking node's value is above 1, it will use more energy than the nodes it is attacking. The attacks indicated in red are those that can be prevented by simple adjustments to the algorithm's implementation.

The figure does show the relation between the different attacks, which should remain mostly the same if used under the same circumstances. For example, in Figure 7.1(c), Sinkhole 3 will be much more effective and use fewer messages than Sinkhole 2, and a HELLO flood 2 will use the same amount of resources as HELLO flood 1, but be far more effective.

It can also be seen that some attacks, although possible, are not very useful, such as the Sybil attack in both Multihop en Mintroute. Also, a normal HELLO flood attack, which needs special hardware and more energy per message, does not perform better in Multihop or Mintroute than for example the sinkhole attack that can be performed by a normal node. Therefore, these weaknesses do not need as much attention as the modified sinkhole and HELLO flood attacks that can have a large impact on the network with only few messages.



(a) Multihop.

(b) Mintroute.

(c) DSDV.

(d) TinyAODV.

Figure 7.1. Effect and required resources of attacks on different routing algorithms.

The figure also shows that routing algorithms using a hop count metric, which are Multihop and DSDV with a hop count metric, can be disrupted more easily than those with a quality metric, which are Mintroute and DSDV with a quality metric. This means that choosing a routing algorithm using a quality metric would be a good choice with respect to added robustness of the network.

It is clear that all studied routing algorithms are insecure but can be made safer when some of the obvious flaws or weaknesses are removed or prevented.

Chapter 8

Conclusions and Future Work

In this thesis we have studied how secure or insecure present day WSNs are. This was done by performing various attacks on hardware and software that is regularly used in WSNs in both research and civil applications as well as in other areas.

8.1 Conclusions

Experiments showed that there are vulnerabilities at all network layers. Therefore, if one wants to have a secure WSN one needs to take security into account from the beginning. This means that protection against jamming and collisions needs to be taken into account when choosing the hardware. As was shown in Chapter 4, the only effective countermeasure against jamming was frequency hopping or UWB, which both need to be supported by the used hardware. The same applies to collisions, which are both hard to detect and hard to counter. As long as messages are sent over a single frequency, the internal structure or used algorithm is unimportant because the bytes themselves can be destroyed by a collision.

Having attack resistant hardware does not mean the security of the software can be ignored. Especially not with the vulnerable routing algorithms that are needed in large WSNs, where nodes need to send their messages through other nodes to reach the other side of the network.

The routing algorithms all show several security flaws, but some of these can be easily prevented with minor adjustments thereby removing some of the more severe attacks.

All layers that have been examined were found to be insecure, but for a large part this is because they were designed without security considerations. In that respect there is a lot of room for improvement.

8.2 Future Work

It is clear that WSNs are still nowhere near secure. To attain better algorithms and security measures, improvements should come from two sides. On one side

one has the creation and research towards mechanisms, algorithms and protocols that are resilient against attacks. For example simple routing cycle checks could be incorporated into the design from the start. Future work could focus on finding simple mechanisms that can be applied to any routing algorithm design that counters the most common weaknesses.

On the other hand one has the different WSN attacks. Knowing how attacks work and why they work can be valuable in finding defences against them. Creating a standard list of attacks that can be performed on a WSN for testing purposes could help make it more secure before deployment.

In this thesis, the research on routing protocol attacks was done in a simulator. It could be useful to see what happens if these attacks are applied to a real WSN and find out what differences arise.

Bibliography

[1] H. Berghel. Wireless infidelity I: war driving. *Communications of the ACM*, 47(9):21–26, 2004.

[2] H. Berghel and J. Uecker. Wireless infidelity II: airjacking. *Communications of the ACM*, 47(12):15–20, 2004.

[3] Berkeley. TinyOS, 2000. `http://www.tinyos.net`.

[4] E. Biham, A. Biryukov, O. Dunkelman, E. Richardson, and A. Shamir. Initial Observations on Skipjack: Cryptanalysis of Skipjack-3XOR. In *SAC '98: Proceedings of the Selected Areas in Cryptography*, pages 362–376, London, UK, 1999. Springer-Verlag.

[5] E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. *Lecture Notes in Computer Science*, 1592:12–23, 1999.

[6] J. Deng, R. Han, and S. Mishra. INSENS: Intrusion-tolerant routing in wireless sensor networks. In *In Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems*, 2003.

[7] D. Ganesan, R. Govindan, S. Shenker, and D. Estri. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review*, 4(5):11–25, October 2001.

[8] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language: A holistic approach to networked embedded systems. In *PLDI '03: Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, pages 1–11, San Diego, California, USA, 2003. ACM Press.

[9] C. Karlof, N. Sastry, and D. Wagner. TinySec: a link layer security architecture for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 162–175, Baltimore, Maryland, USA, 2004. ACM Press.

[10] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, 1(2–3):293–315, September 2003.

[11] M. Kershaw. Kismet. `http://www.kismetwireless.net`.

[12] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137, Los Angeles, California, USA, 2003. ACM Press.

[13] H. Liu, H. Ma, M. El Zarki, and S. Gupta. Error control schemes for networks: an overview. *Mobile Network Applications*, 2(2):167–182, 1997.

[14] D. Malan. Crypto for tiny objects. Technical report, Computer Science Group Harvard University Cambridge, Massachusetts, April 2004.

[15] V. Moen, H. Raddum, and K. J. Hole. Weaknesses in the temporal key hash of WPA. *SIGMOBILE Mobile Computing and Communications Review*, 8(2):76–83, 2004.

[16] T. Park and K. G. Shin. Lisp: A lightweight security protocol for wireless sensor networks. *Transactions on Embedded Computing Systems*, 3(3):634–660, 2004.

[17] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[18] C. Perkins and E. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, Louisiana, USA, 1999.

[19] A. Stubblefield, J. Ioannidis, and A. Rubin. A key recovery attack on the 802.11b wired equivalent privacy protocol (WEP). *ACM Transactions on Information and System Security*, 7(2):319–332, 2004.

[20] M. Tham. Exponentially weighted moving average filter. `http://lorien.ncl.ac.uk/ming/filter/filewma.htm`.

[21] R. Watro, D. Kong, S. 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*, pages 59–64, Washington DC, USA, 2004. ACM Press.

[22] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 14–27, Los Angeles, California, USA, 2003. ACM Press.

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