Alzaid, Hani and Abanmi, Suhail and Kanhere, Salil and Chou, Chun Tung (2008) BANAID: A Sensor Network Test-bed for Wormhole Attacks . In *Proceedings AusCERT 2008*, Gold Coast, Australia.

BANAID: A Sensor Network Test-bed for Wormhole Attacks

Hani Alzaid
Information Security Institute
Queensland University of Technology
Australia
halzaid@isi.qut.edu.au

Suahail Abanmi , Salil Kanhere , Chun Tung Chou
School of Computer Science and Engineering
University of New South Wales
Australia
{abanmis , salilk , ctchou} @cse.unsw.edu.au

Abstract

The development of wireless sensor devices in terms of low power and inexpensive data-relaying has been partially achieved because of the rapid progress in integrated circuits and radio transceiver designs and device technology. Due to these achievements, the wireless sensor devices are able to gather information, process them if required, and send them to the next sensor device. In some applications, these wireless sensor devices must be secured, especially when the captured information is valuable, sensitive or for military usage. Wormhole attacks are a significant type of security attacks which can damage the wireless sensor networks if they go undetected. Unfortunately, these attacks are still possible, even if the communication is secured. The wormhole attack records packets at one point of the network, passes them into another node and this last node injects the packet into the wireless sensor network again. This type of attacks can not be avoided by using cryptographic techniques because attackers neither generate new nor alter existing packets. They only forward legitimate packets from part of the network to another part. This attack can cause damage to the route discovery mechanism used in many routing schemes. In this paper we build an actual test bed, which is called BANAID, to simulate the wormhole attack on a wireless sensor network and then implement one of the current solutions against this attack. BANAID consists of a combination of Mica2 motes and Stargate sensor devices.

1. Introduction

The capability of combining sensing, processing, and communicating wirelessly have been enabled by the advances in microelectronics fabrication [2]. A Wireless Sensor Network (WSN) is composed of a group of tiny sensor devices, which can be networked together and deployed in a wide spectrum of applications in various military and civil domains. The main objectives of deploying the Wireless Sensor Network (WSNs) are remote monitoring and gathering information [2]. Most of WSN's applications run in non-trusted environments and require secure communications such as emergency response operations, military or police networks, and safety-critical business operations. For example, in emergency response operations such as after a natural disaster like flood, tornado, or earthquake, a wireless sensor network could be used for real time feedback. So, emergency rescue will relay on that particular type of networks [3].

Unfortunately, this type of network is vulnerable to several attacks. One major type of these attacks is known as a Wormhole attack where an attacker records a packet at one location in the network, tunnels the packet to another location, and replays it at other part of the network [3]. The wormhole attack places the attacker in a very powerful position, allowing him to gain unauthorized access, disrupt routing, or perform a Denial-of-Service (DoS) attack [3]. Current solutions for Wormhole attack such as [4, 9,10] are evaluated by running the proposed techniques in different simulations. There is no real deployment for any of these solutions. To the best of our knowledge, this is the first work that built an actual test bed to show the visibility of the Wormhole attack in WSN. BANAID consists of few numbers of Mica2 [1] and two Stargate [2] sensor devices. Moreover, one of the proposed solutions, which is Packet Leashes [4], will be implemented.

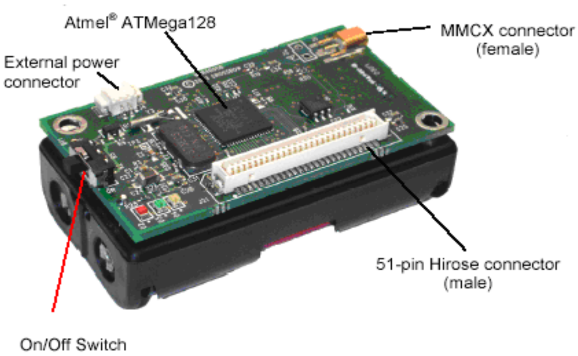

Figure 1: MICA2 Motes without an antenna [1].

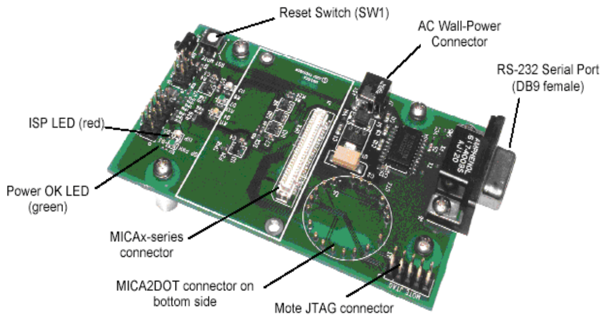

Figure 2: MIB510 Serial Interface Board [1].

The rest of the paper is organized as follows. In Section 2, an introduction about different components used in building up BANAID will be given. Section 3 describes the designing and the implementation of BANAID. Section 4 concludes the paper.

2. BANAID Components

It is important to have some background about the wireless sensor devices used in this paper to build up the test bed. BANAID consists of several Mica2 motes and two Stargate sensor devices. In the following subsections, brief information about these two types of wireless devices is given.

2.1 MICA2

There are different models of Motes that have been produced by Crossbow[1]. Each of these models has different features. These models are MICAz, MICA2, MICA2DOT, and MICA. All models except MICA2 are beyond the scope of this paper. MICA2 Motes have been used in this paper to build BANAID. Therefore, the features, hardware layouts, and software environment of MICA2 Motes are described in the following subsections.

2.1.1 MICA2 Features

The MICA2 Motes come in three types according to their RF frequency band: the MPR400 (915 MHz), MPR410 (433 MHz), and MPR420 (315 MHz). The Motes use the Chipcon CC1000, FSK modulated radio. All types utilize a powerful Atmega128L micro-controller and a frequency tunable radio with extended range. The MPR4x0 and MPR5x0 radios are compatible and can communicate with each other. (The x = 0, 1, or 2 depending on the type / frequency band) [1]. Figure 1 shows a sample of MICA2 Mote. The current version of Mica2 uses a 16 bit, 8MHz Texas Instruments, 1024 KB external flash, and is powered by two AA batteries.

2.1.2 Hardware Layout

The MICA2 Mote can be reprogrammed using an external board called MIB510 Serial Interface Board. This board is a multi-purpose interface board used with MICA, MICA2, MICAz, and MICA2DOT Motes family. It supplies power to the devices through an external power adapter option, and provides an interface for a RS-232 Mote serial port and reprogramming port [4]. The MIB510 serial interface board, as shown in Figure 2, is used to program the MICA2 Mote. This board has the PC connection capability using the RS-232 serial port. Programming the Motes requires a special operating system called TinyOS, which should be installed in the PC.

2.1.3 Software Environment

MICA2 Motes uses a special operating system, which is used for wireless sensor nodes, called TinyOS [7]. This operating system is an open-source event-driven operating system designed for wireless embedded sensor networks. It features a component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks. TinyOS's component library includes network protocols, distributed services, sensor drivers, and data acquisition tools – all of which can be used as-is or can be further refined for a custom application. TinyOS's event-driven execution model enables fine-grained power management, yet allows the scheduling flexibility made necessary by the unpredictable nature of wireless communication and physical world interfaces [1].

[1] http://www.xbow.com/Home/HomePage.aspx

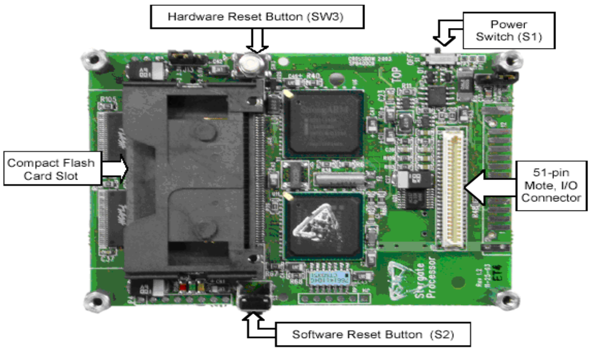

Figure 3: Processor Board (Top View)[2].

TinyOS have been implemented in a language called nesC. This language is an extension to C which has been designed to embody the structuring concepts and execution model of TinyOS. Programs written in nesC language are built out of components, which are wired to form entire programs. Each component has interfaces which can provide its functionality to other users.

2.2 Stargate

Stargate is a powerful single board computer with enhanced communications and sensor signal processing capabilities. This product was designed within Intel's Ubiquitous Computing Research Program, and licensed to Crossbow for production. In addition to traditional single board computer applications, the Stargate directly supports applications around Intel's Open-Source Robotics initiative as well as TinyOS based Wireless Sensor Networks.

2.2.1 Stargate Features

Stargate uses Intel's latest generation 400 MHz XScale processor (PXA255) and SA1111 StrongARM companion chip for I/O access. It has a reset button, real time clock, lithium ion battery option and 51-pin daughter card interface [2]. The Stargate sensor device used in this test bed has various features which can also be used in different applications. The main feature that has been used extensively in the test bed is the Compact Flash slot. Stargate has the capability to have a WiFi Compact Flash card. Another expansion in Stargate is MICA2 Mote connector, which allows the Stargate to communicate with other MICA2 Motes through the radio channel. Stargate consists of two hardware pieces: the processor board and the daughter card. These pieces will be explained in details in the following subsection

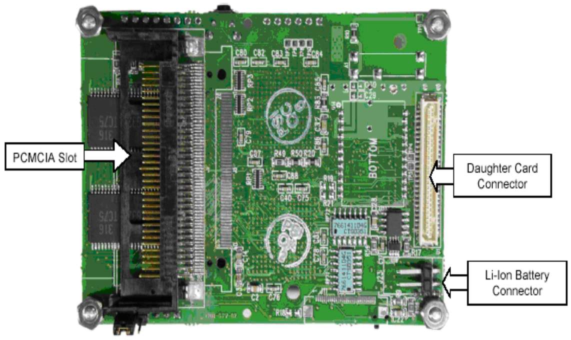

Figure 4: Processor Board (Bottom View) [2].

2.2.2 Hardware Layout

As mentioned in the previously, Stargate consists of two hardware pieces: a processor board and a daughter board. Each of these pieces will be described in this subsection. The processing board as appeared in Figure 3 and Figure 4 shows all the main buttons and slots. Figure 3 shows the top view of the processing board. It is clear from this view that

Stargate has two slots. These slots can be used to communicate with other devices. The first slot is used to allow Stargate to communicate with other MICA2 Mote by connecting a MICA2 Mote over the Stargate. The second slot is used to connect WiFi Compact Flash card to the Stargate, which allow it to communicate through the standard 802.11a or 802.11b wireless protocols.

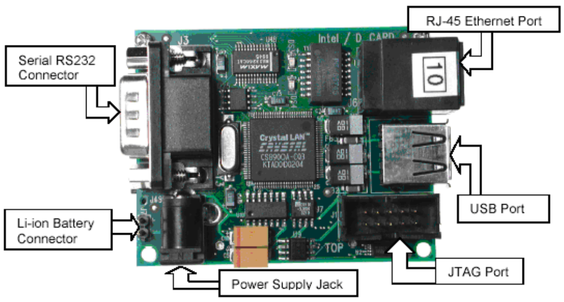

Figure 5: Daughter Card (Top View) [2].

The processor board gets its power from the daughter card. The daughter card has a power supply jack as it appears in Figure 5 and Figure 6. Also, the daughter card allows the user to communicate with Stargate using different types of interfaces. There are three different ways to communicate with Stargate. The first one is using Serial RS232 Connecter.

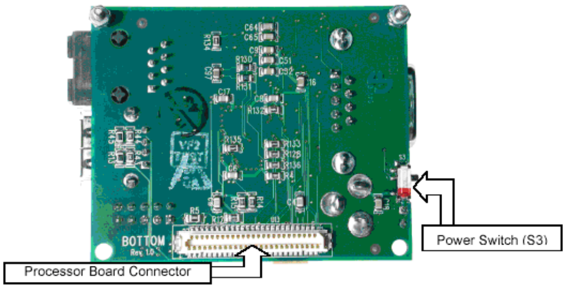

Figure 6: Daughter Card (Bottom View) [2].

The second is using RJ-45 Ethernet Port. And the third one is using USB Port. All these interfaces give the ability to control or upload programs to the Stargate. All switch buttons in the Stargate processor board and daughter card have to be switched on before using Stargate. There are two switch buttons S1 and S2 in the processor board. The third switch button S3 is located in the daughter card.

2.2.3 Software Environment

As mentioned before, the Stargate is using an embedded Linux operating system kernel. It is installed on the processing board of the Stargate. There is also additional software shipped with the Stargate development platform, which could be used to enable program development. The Stargate's platform provides the capability of installing programs written in C language. The developer can control various functions in Stargate by using C language programs after compiling and installing them.

3. Building BANAID

In this section the design, implementation, explanation of the wormhole attack, and the implementation of its solution will be described.

3.1 Design of BANAID

The proposed test bed is developed with the following underlying assumptions:

- The chosen network topology is assumed to be fixed.
- Each node is assumed to know its neighbors.

Figure 7 shows that BANAID is composed of seven Mica2 sensor nodes and two Stargate devices. In this network design, the original source is Mote 1 and the original destination is Mote 4. The global clock is a normal sensor (Mote) which keeps sending a clock packet to synchronize all other motes.

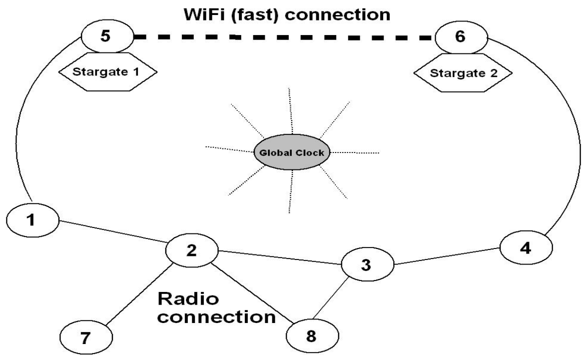

Figure 7: The proposed design of BANAID.

Each Mote in each Stargate will only forward any received message from radio connection to WiFi connection and vice versa without changing these messages. The routing algorithm that has been used in BANAID is the Ad hoc On Demand Distance Vector (AODV). This routing algorithm will be explained in the next subsection.

Moreover, the wormhole attack happens during the phase of building the routes between nodes in AODV. The attack will affect the routing table entries in the original source and destination motes. Therefore, the actual path that a message should pass from the original source to the original destination will be imprecise. The impact of the wormhole attack on the network will appear clearly in both the original source and the original destination motes. The original source mote will deal with the original destination mote as its direct neighbor and vice versa, which is not right because they are separated by two intermediate motes.

3.2 The AODV Algorithm

The Ad hoc On Demand Distance Vector (AODV) routing algorithm is a routing protocol designed for ad hoc mobile networks. It is an on demand algorithm, meaning that it builds routes between nodes only as desired by source nodes. It maintains these routes as long as they are needed by the sources.

AODV builds routes using route request and reply query messages. When a source node desires a route to a destination for which it does not already have a route, it broadcasts a route request (RREQ) packet across the network. Nodes receiving this packet update their information for the source node and set up backwards pointers to the source node in the route tables. A node receiving the RREQ may send a route reply (RREP) if it is either the destination or if it has a route to the destination and the sequence number in RREQ packet is not present in the node. If this is the case, it sends a RREP back to the source. Otherwise, it forwards the RREQ. Nodes keep track of the RREQ's packets by storing their sequence numbers. If they receive a RREQ which they have already processed, they discard the RREQ and do not forward it.

As the RREP propagates back to the source nodes set up forward pointers to the destination. Once the source node receives the RREP, it may begin to forward data packets to the destination. The routing table for each node will be updated according to the hop count field in RREQ or RREP packets. The received packet with the smallest hop count will be chosen as the best route path.

However, in this paper, not all of AODV routing algorithm features have been implemented, features that meet the aim of this paper are just implemented for the sake of simplicity. For example, features like sending and receiving RREQ/RREP packets, and building the routing tables for each node have been implemented whereas features like maintaining the route paths and sending hello messages have not been implemented.

3.3 Implementation of BANAID

According to the proposed design of BANAID, there are seven MICA2 Motes and two Stargate sensor devices. Two motes have been combined with the two Stargate sensor devices which are mote 5 and mote 6.

To implement BANAID, three types of programs have been written and installed in the motes and the stargate sensor devices. The first program has been installed in motes 1, 2, 3, and 4, which is a simple customized AODV algorithm. The second program has been installed in motes 5 and 6, which forwards any received packet from the radio antenna to the serial port that connects the mote with the Stargate and vice versa, without changing this packet. The third program has been installed in both Stargate 1 and 2. When a packet is received by the Stargate from its serial port, which is connected to the corresponding mote, the program forwards it to the other Stargate through its WiFi connection and vice versa. The other Stargate will receive this packet from its WiFi connection and forward it to its serial port, which is also connected to the corresponding mote. Then this packet will be broadcasted via the radio antenna of the mote.

3.3.1 The AODV Program

Before describing the functionalities that are done by this program, a brief explanation about the important variables is given as follows:

OS	OD	C	N	HC	MID	MT	S/M	TS

Table 1: Packet Format.

Table 1 illustrates the packet format used in building this test bed where

- **OS** represents original sender address.
- **OD** represents original destination address.
- **C** represents the current sensor address.
- **N** represents the next sensor's address.
- **HC** represents the hop counter.
- **MID** represents the messages ID.
- **MT** represents the message type.
- **S/M** explains whether the message comes from a Stargate or a Mica2 sensor.
- **TS** represent time stamp.

Dest. Address	Next Hop	Hop Count
…		

Table 2: Routing Table Format.

Next, Table 2 describes the routing table format. It is a k*3 array where *k* is a predefined value that represents the maximum number of entries that the routing table can hold. The table consists of three fields: the *destination address* field which stores the final destination for the current received packet, the *next hop* field which holds the address for next hop, and the *hop count* field which represents the number of hops left before reaching the final destination.

Figure 8 illustrates the pseudo code[2] that has been used in this paper to implement AODV. It describes the AODV's functionalities that have been implemented in order to build this test bed. It does the most important part of this paper since it implements the AODV routing algorithm, synchronizes with global clock, and implements the a solution to the wormhole attack. This program has been written in combination of nesC and C languages and will be upload into Mote 1,2,3, and 4.

[2] The whole program can be obtained by contacting the first author.

Algorithm 1: AODV Program

```
intitilization phase;
send PKT;
run server thread with IP:10.1.1.2;
synchronize internal clock with the global clock;
if PKT_MSG_Type == 0 then                         // RREQ mode
    if PKT_MSG_ID is not in RREQ list then
        add entry into Routing Table;
        add PKT_MSG_ID into RREQ list;
        if PKT_Orign_Destin == Current_Mote_Address then
            prepare RREP PKT and send it;
        else
            forward PKT;
        end if

    else
        check RT for PKT_Orign_Source and get the Hop_Count;
        if Hop_Count > PKT_Hop_Count then
            update the entry in Routing Table;
            send PKT;
        end if
    end if

else if PKT_MSG_Type == 1 then                    // RREP mode
    if PKT_MSG_ID is not in RREP list then
        add entry into Routing Table;
        add PKT_MSG_ID into RREP list;
        if PKT_Orign_Destin == Current_Mote_Address then
            prepare RREP PKT and send it;
            send routing information for this mote;
        else
            forward PKT;
        end if

    else
        check RT for PKT_Orign_Source and get the Hop_Count;
        if Hop_Count > PKT_Hop_Count then
            update the entry in Routing Table;
            send PKT;
        end if
    end if
else if PKT_MSG_Type == 2 then
    if PKT_Orign_Destin == Current_Mote_Address then
        packet reaches its destination;
    else
        check RT for PKT_Orign_Source and get Next_Hop;
        send PKT for the next hop;
    end if
end if
```

Figure 8: Pseudo Code of AODV.

3.3.2 TOSBase Program

This program was written and delivered with the TinyOS tool kit as one of many readymade applications. The name of this application is TOSBase, which can be found under the application directory in TinyOS file. It simply forwards any received packet from the radio antenna of the Mica2 sensor to the serial port (51-pin Hirose Connector), which is shown in Figure 1, without changing the content of the packet. The Mica2 sensor is connected to the Stargate through this serial port. This program is written in nesC language and will be uploaded into Mote 5, and 6 (see Figure 7).

3.3.3 Stargate Program

The program consists of two threads or processes[3]. The first process is the main program, which keeps listening to the serial port connected to the mote and sends any received packet to the client socket.

[3] The whole program can be obtained by contacting the first author.

Algorithm 3: Stargate Program

```
A: Stargate One's Program
intitilization phase;
open serial port;
run server thread with IP:10.1.1.2;
// the previous command runs forever to receive any
// msg from Stargate2 with IP:10.1.1.3 and forward it
// to the serial port to be sent by Mica2 sensor via radio
for (1) do
    if Stargate receive a msg from the serial then
        if msg comes from mote 1 and it is RREQ then
            call client to forward the msg through WiFi to Stargate2;
        end if
    end if
end for

B: Stargate Two's Program
intitilization phase;
open serial port;
run server thread with IP:10.1.1.3;
// it runs forever to receive any msg from Stargate1 with
// IP:10.1.1.2 and forward it
// to the serial port to be sent by Mica2 sensor via radio
for (1) do
    if Stargate receive a msg from the serial then
        if msg comes from mote 4 and it is RREP then
            call client to forward the msg through WiFi to Stargate1;
        end if
    end if
end for
```

Figure 9: Pseudo Code of Stargate's Program

The client socket will initiate a WiFi connection with the server socket in the other Stargate. The second thread or process runs as a server that keeps listening to any client connection via WiFi. It receives the packet and forwards it to the serial port connected to the mote, which will broadcast the packet via its radio antenna. As explained in Figure 9, the program in Stargate 1 will only forward messages from radio that comes from the original source which is mote 1. The other Stargate 2 will only forward messages from radio that comes from the original destination mote 4. The two versions of these programs are the same except in checking the type of the message before start forwarding it.

3.4 Demonstration of BANAID

This subsection illustrates the usability of this test bed. The Mica2 Motes have some indicators which have been used to simulate this type of attack. Each mote has three lights: red, green, and yellow. These lights have been used in this paper to simplify the demonstration of this solution such as showing how packets are traveling from one mote to another and showing when and how the wormhole attack happens. For example when the green light blinks, it means that the mote is sending a packet. The red light indicates that the mote has received a RREQ packet from other motes. The yellow light turns on when the mote has received a RREP packet from other motes. The green and yellow lights together indicate that a RREQ or RREP has been received at the destination. It is clear to say that the red light will be seen before the yellow light since the request packet comes before the reply. However, if the yellow light has been seen before the green light, it means a wormhole attack has been detected. The reason for this is the replay comes before or faster than the request packet. By doing this, it is easier now to understand what is happening during the demonstration.

3.5 Visibility of BANAID

There are two situations to run BANAID. The first situation is demonstrating the customized AODV routing algorithm without implementing the wormhole attacks. In other words, demonstrating customized AODV without the Stargate sensor devices. The second situation is demonstrating wormhole attack beside the customized AODV.

To demonstrate the first situation, it is important to turn the Stargate sensor devices off. The wormhole attack will not occur in this situation. To start this demonstration, firstly turn on motes 2, 3, 4, and 7. Then, turn on the mote 1 which acts as the original source. Mote 1 will start sending RREQ packet by blinking its green light. This sending will be after synchronized with global clock. Mote 2 will receive the RREQ packet, turn on its red light, and forward this packet by blinking its green light. Mote 3 will do the same thing that mote 2 has done. Finally, mote 4 will receive the RREQ packet, turn on its red light and yellow light indicating that the RREQ has reached its destination, and prepare the RREP packet. The same process will happen for RREP packet, but the difference is that each mote will turn on its yellow light on receiving the RREP packet. Finally, Mote 4 will turn on its red and yellow lights on receiving the RREP packet indicating that the RREP has reached its destination.

The other situation is demonstrating the wormhole attack by adding the two Stargate with their motes. Firstly, turn on all motes and Stargates except mote 1 which is the original source. Then, turn on mote 1 to start sending the RREQ packet. In this situation, mote 4 which is the original destination will receive the RREQ from mote 1 directly through the Stargate path since the WiFi path is faster than the radio path. This happens when Mote 5 receives the RREQ packet from mote 1 and sends it directly through the WiFi connection to the other Stargate. Mote 6 in the other Stargate will send this packet to mote 4 via its radio antenna. The RREQ packet will travel through the WiFi connection faster than the normal path which is through the intermediate motes 2 and 3. In this situation, mote 4 will think that mote 1 is its direct neighbor and it will turn on its red light and yellow light preparing for the RREP packet. Mote 4 will send the RREP packet which will be delivered through the WiFi connection between the two Stargates. Mote 1 will turn on its red and yellow lights when it receives the RREP packet. Also, mote 1 will think that mote 4 is its direct neighbor and this is how the wormhole attack happens. It is possible to display the information of the packets and the final routing table in mote 1 by using a PC. More details of how to perform that have been provided in Chapter 3.

There are a number of techniques to solve the wormhole attack in a wireless sensor network. In the next section, some of these techniques will be briefly described. The implementation of one of them will be explained in section 3.7.

3.6 Wormhole Attack Solution

To detect and deter the Wormhole attack, some solutions have been proposed in different papers such as [4,9,10]. Hu et al. proposed a defense against the Wormhole attacks in WSN called packet leashes [4]. It is the most commonly cited solution and therefore it is the only solution that is implemented in this paper. A leash is a portion of information that is added to the packet to restrict its traveling distance or time. This solution consists of two types of leashes: geographic leashes and temporal leashes. Each of these types will be described briefly in this section and more details are available in [4].

A geographic leash detects and prevents the wormhole attack by ensuring that the sender and the receiver are within a specified distance. To do that, each node must know its location and be timely synchronized with other nodes. The other type of solution is a temporal leash. It detects and prevents the wormhole attack by ensuring that the packet's traveling time is within a specified period of time. To do that, all nodes must be timely synchronized in terms of their clocks. When the sender starts sending the packets, it stores its sending timestamp in the packet. Then, the receiver can compare its receiving timestamp with the value in the packet. Therefore, the receiver will be able to detect if the packet traveled so fast according to a specified transmission time. In BANAID, the temporal leash solution has been used and implemented to detect and prevent the wormhole attack in the wireless sensor network. The geographic leash solution has not been considered in this paper because it is complicated to be implemented. It needs a special hardware that can specify the locations of all nodes such as GPS, which is expensive.

3.7 Implementation of the 'Packet Leashes Solutions'

As it has been discussed earlier, timing mechanism is required to distinguish if the packet received from the fake path or the real path to avoid the wormhole attack. A packet received from the Stargate is going to be faster than the packet received from the mote since the WiFi transmission is faster than the radio transmission. Therefore, applying this timing mechanism will allow the destination to recognize the real packet by comparing it's time stamp with the time stamp of the received packet. If the difference between these two time stamps is less than or equal to X, it means that it is a fake packet. To compute the value of X according to BANAID design:

- There is a transmission delay in each mote. The average transmission delay is around 45 msec(Tr = 45 msec).[5].
- The real path consists of mote 1, 2, 3, and 4 where the fake path consists of mote1, 5, 6, and 4.
- Processing time at each mote will be neglected since both paths have the same number of motes. Therefore, the total processing time for motes in each path will be almost same.
- The WiFi transmission delay is neglected since the data rate is high.

Delay on the real path:
= No. of motes that will transmit * Tr
= 3 (mote1, 2, and 3) * 45
= 3 Tr.

Delay on the fake path:
= No. of motes that will do transmission * Tr
= 2 (mote1, and 6) * 45
= 2 Tr.

Therefore, the destination node is able to distinguish between the real and the fake packet by subtracting the time stamp at destination by the time stamp at the source. If the result <= 2 Tr, then the packet is fake. Otherwise, it is real.

4. Conclusion and Future Work

The rapid development in the wireless sensor networks field has allowed this technology to be used in many applications. Some of these applications are critical and require secure and trusted environment. Therefore, different research studies have been conducted to analyze the wireless sensor networks and discovering their threats. One of the attacks which may damage wireless sensor networks is the Wormhole attack. Thus, this paper built a test bed of a group of sensor devices to simulate the wormhole attack and implement one of the solutions to detect and prevent this attack. In this paper, a brief background about the types of wireless devices, which are used in building up BANAID, is introduced. Then, the design and the implementation of BANAID are illustrated by describing the proposed network topology, the algorithms, and the chosen solution. Finally, detailed steps of setting up BANAID are given. In future, the timing mechanism that has been used might be improved especially with the new version of TinyOS that give the opportunity to implement internal clock in the mote. Also, BANAID could be enhanced by some additional improvements such as implementing and evaluating different solutions for the Wormhole attack, and extending the size of the size of the network implemented in BANAID.

Acknowledgement

The authors would like to acknowledge Dr. Wen Hu, CSE, UNSW for his support with setting environment for TinyOS, Mica2, and Stargae. Also, the authors would like to acknowledge Manal Alfaraj, Griffith University and Teresa Desmarchelier, University of Queensland for the valuable feedback on the paper. This work was supported by King Abdulaziz City for Science and Technology (KACST).

5. References

[1] Crossbow Technology, Inc., MPR/MIB Mote Hardware Users Manual, http://www.xbow.com/Support/Support_pdf_files/MPR-MIB_Series_Users_Manual.pdf

[2] Crossbow Technology, Inc., Stargate Developer's Manual, http://www.xbow.com/Support/Support_pdf_files/Stargate_Manual.pdf

[3] Ganeriwal, S., Kumar, R., & Srivastava, M.B. "Timing-sync protocol for sensor networks," ACM Conference on Embedded Networked Sensor Systems (SENSYS 2003).

[4] Hu, Y., Perrig, A., & Johnson, D.B. "Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks," Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003), vol. 3, pp. 1976-1986, IEEE, San Francisco, CA, April 2003.

[5] Paek, J., Chintalapudi, K., & Govindan, R."A Wireless Sensor Network for Structural Health Monitoring: Performance and Experience"2005.

[6] Stargate Mailing List, http://www.cens.ucla.edu/pipermail/stargate-users

[7] TinyOS website, http://www.tinyos.net/

[8] TinyOS Mailing List, http://www.tinyos.net/search.html http://www.cens.ucla.edu/pipermail/stargate-user/

[9] Lazos, L.; Poovendran, R.; Meadows, C.; Syverson, P.; Chang, L.W., "Preventing wormhole attacks on wireless ad hoc networks: a graph theoretic approach," Wireless Communications and Networking Conference, 2005 IEEE , vol.2, no., pp. 1193-1199 Vol. 2, 13-17 March 2005.

[10] Khalil, I.; Saurabh Bagchi; Shroff, N.B., "LITEWORP: a lightweight countermeasure for the wormhole attack in multihop wireless networks," Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on, vol., no., pp. 612-621, 28 June-1 July 2005.