THE UNIVERSITY OF NEW SOUTH WALES

SCHOOL OF ELECTRICAL ENGINEERING
AND
TELECOMMUNICATIONS

Access Control using Bluetooth

TM09

Mei, Jiexiang Marvyn - 2244246
Salim, Agus - 2251992

Supervisor: Dr. Tim Moors
Assessor: Dr. Saeid Nooshabadi

November, 2003. Sydney, Australia.
Bachelor of Engineering in Electrical Engineering

Acknowledgements

Both members of this group thesis would like to thank a few people for their kind help and support throughout the duration of this project.

Firstly, we would like to thank our thesis supervisor, Dr Tim Moors, who not only responded to the problems our group faced during the thesis, but has also done his best to make available whatever resources and logistics support we required, in a timely manner.

We would also like to thank Joseph Yeo, the laboratory technician in-charge of the Bluetooth laboratory, for providing us with the required logistics support we required, including, but not limited to, making available to us the computers that were crucial to the development of the software in this thesis, and also the servicing of computers that broke down.

Abstract

This thesis project proposes a method of access control using Bluetooth. Currently access control methods require physical contact to a device, such as a swipe-card. Bluetooth is a wireless technology that could be used to replace such applications, and provide the convenience of wireless access control. The purpose of this thesis is to demonstrate that such a concept is feasible, by implementing one such application.

The application developed in this thesis involves the shutting off of ringing tones of mobile phones that enter a “silent” zone. Such a policy is enforced by a Bluetooth access point situated in this “silent” zone. The access point will send out a “beacon” signal to the mobile phone, via Bluetooth, telling it to shut down its ringing tone. Another focus of this thesis is to ensure that this procedure is secure. Both the mobile phone and access point will have to carry out an authentication procedure, designed in this thesis.

With the success of this thesis, such a concept could be extended to other forms of access control applications.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

The Bluetooth standard and technology came about initially when Ericsson Mobile Communications carried out a study to find a low power and low cost radio interface between mobile phones and their accessories. The study showed that a short-range radio link solution was feasible. To develop the technique and to get broad market support, Ericsson, together with Intel, IBM, Toshiba and Nokia Mobile Phones formed a Special Interest Group (SIG) in 1998. This group was to form a standard for the air interface and the software that controls it, such as to achieve interoperability between different devices from different producers. [1]

1.2 Motivation

The motivation of this thesis is to demonstrate the use of Bluetooth technology in access control applications. One particular application would be to turn off the ringing tone of a Bluetooth-enabled mobile phone as it enters a lecture hall, as the ringing of the mobile phone is undesirable in that location.

Currently, most access control solutions are implemented via the use of conventional technologies, such as bar-coded swipe cards, and access pin code numbers. With the introduction of newer mobile devices with Bluetooth, it is possible to replace older access control technology with applications developed using the Bluetooth technology, allowing access control to be delivered in a wireless fashion.

Because Bluetooth is a relatively new technology, not many Bluetooth applications have been created. Hence writing a Bluetooth software system would be a challenging, but interesting experience for our final year thesis.

2. Features of Bluetooth

2.1 Strengths

The following table compares the Bluetooth radio to wireless LAN and infrared. These three technologies are the most commonly used in many of today's wireless applications. Each of them has their own set of advantages and disadvantages, and this makes each of them suitable to certain applications.

	Bluetooth	Wireless LAN	Infrared
Typical Range	Medium (10 m)	Long (100 m)	Short (1 m)
Line-of-sight	No	No	Yes
Bandwidth	1 Mbps shared	11 Mbps shared	115 kbps & 4 Mbps dedicated
Interference	Other RF devices	Other RF devices	None
Security	Less secure than infrared. Uses link-layer authentication. Still requires application layer security.	Insecure unless protected. e.g. WEP & WPA encryption	Very secure, due to short range and line-of-sight requirement
Power Consumption	High. Needs to maintain a connection	Very high. Needs to maintain a connection.	Low. No constant connection like wireless radios.
Component Cost	About $20, expected to drop to $5.	About $25.	Less than $2.

Table 1: Comparison of Bluetooth, wireless LAN and infra-red technologies. Source: [2]

The main features of Bluetooth that makes it suitable for use with our project are:

- Minimal hardware dimensions.
- Low price of Bluetooth components.
- Low power consumption for Bluetooth connections.
- Inherent security features (described in section 2.6).
- Medium range.

The low cost and small size of the Bluetooth radios means that it can be integrated into many portable devices cheaply. The products offered from companies in the Bluetooth SIG, such as mobile phones, PDAs etc, creates a huge market potential for Bluetooth devices and their applications.

Low power consumption is especially important in this project because the software system requires the Bluetooth radio on the mobile phones to be turned on all the time. This helps to prolong battery life, which is scarce in mobile phones.

The inherent security features and medium range of Bluetooth makes Bluetooth relatively secure as compared to other wireless radios such as wireless LAN. The security features makes it hard to listen to the data transmissions. The medium range means that hackers would have to be within close physical range to the Bluetooth radio in order to listen to its traffic. All these are important, because this project deals with access control, in which security plays a very important role. Note, however, that Bluetooth security is insufficient for this application because it does not stop other Bluetooth devices from "attacking" the system. An additional application-layer authentication procedure will be needed to filter out these malicious Bluetooth users, as described in section 6.5.6.

The relative low data transfer rate of Bluetooth is not a problem because most data transfers in this project will only involve small quantities of data.

2.2 Marketing Aspects

In the last decades, consumer products such as PCs, laptops, personal digital assistants, cell phones etc have increased in popularity. This is based on the continuous cost and size reduction of these devices [3]. The transfer of information between these devices has been hindered because of the need of cables. Bluetooth provides a solution to this by eliminating the need for cabling. It also provides the means for connecting several units to each other, such as setting up small radio Personal Area Networks between any types of Bluetooth devices [1].

2.3 Radio Spectrum

Bluetooth operates on the unlicensed 2.4 GHz spectrum. This means that the spectrum is open to the public without the need for licenses, as long as they meet requirements specified by the FCC. Moreover, Bluetooth applications are targeted at consumers who do a lot of travelling. Hence the spectrum on which Bluetooth operates on must be available worldwide. The 2.4 GHz spectrum is free in most countries of the world and thus meets this criteria.

The use of this spectrum introduces a lot of interference sources to the Bluetooth radio transmissions. One source of interference is high-powered transmitters such as microwave ovens and lighting devices, which also transmit at the 2.4 GHz band. Another source of interference is co-user interference [3], which comes from other Bluetooth users.

Interference immunity can be obtained by interference suppression or avoidance [3]. However these techniques will not be discussed in this report.

2.4 Ad-Hoc Radio Connectivity

The Bluetooth radio system stands out from other radio systems due to its ad-hoc connectivity. The majority of radio systems used today are based on cellular radio architectures. A mobile network is established on a wired backbone infrastructure, and consists of one or more base stations located in different locations to provide cellular coverage. The mobile terminals then access the network in these coverage areas. Hence there is a clear separation between the base stations and the terminals of such systems.

In contrast, there are no distinctive base stations or terminals in the Bluetooth system, nor are there any differences between radio units. There is no wired infrastructure to support connectivity, and there is no predefined central controller for units to rely on for making interconnections [3].

This means that during the implementation of the system, we would not have to worry about the architecture of the resulting network. This is also important because the Bluetooth-enabled devices will be constantly on the move, resulting in a constantly changing network. The lack of an infrastructure means that the network can accommodate such changes easily.

2.5 Network Topologies

Bluetooth devices can be organized into groups of two to eight devices, which together form a piconet. Each piconet will contain at least one master, and all other units participating in the piconet will be slaves. The master of a piconet controls the communications within a piconet [4].

The number of units in a piconet is deliberately limited to eight (one master, seven slaves) in order to keep a high capacity link between all the units. It also limits the overhead required for addressing. Note that the master/slave role only lasts for the duration of the piconet. Once the piconet is cancelled, these roles are also cancelled. Any unit can become master or slave. By definition, the unit that establishes the piconet becomes the master [3].

Two or more piconets can be interconnected to form a scatternet. The connection point between two piconets is a Bluetooth unit that is a member of both piconets. One device can be a master in one piconet, and a slave in another. A device can also be a slave on more than one piconet, but it cannot be a master in more than one piconet, as this will mean that the two are actually one piconet (having a single master).

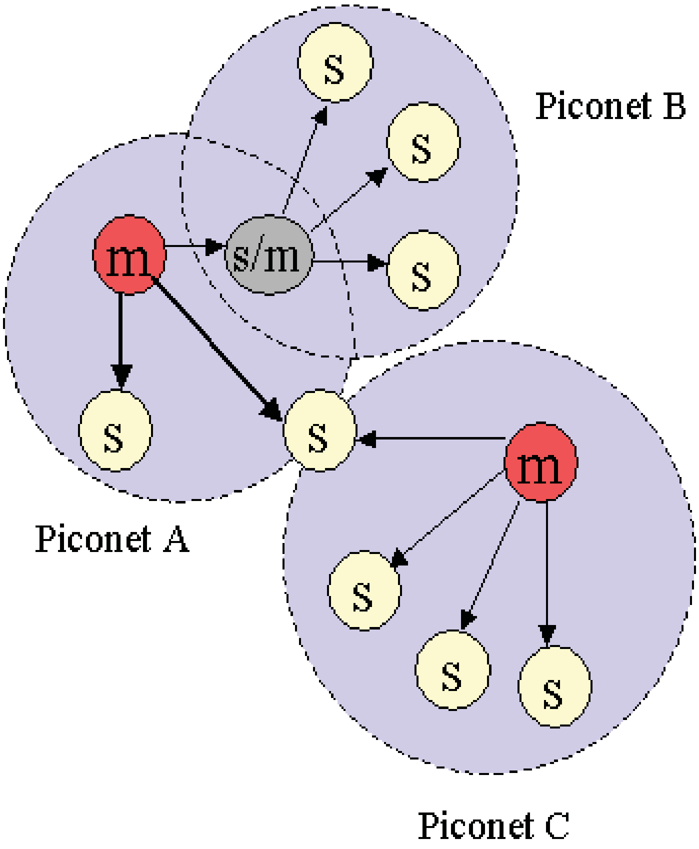


Figure 1: Shows a scatternet comprising of three piconets. Two piconets are linked to each other by a node which exists in both piconets. (Figure adapted from [1].)

2.6 Bluetooth Security

Security in Bluetooth is provided in three ways [1]:

1. Pseudo-random frequency hopping.
2. Authentication
3. Encryption

Frequency hopping is a spread spectrum technique that was intended for noise resilience. However it is also a good way to prevent eavesdropping [5], because it is very hard for an eavesdropper to track the frequency changes of a transmission over a Bluetooth network. The following figure shows how the frequency of a Bluetooth channel can change over time.

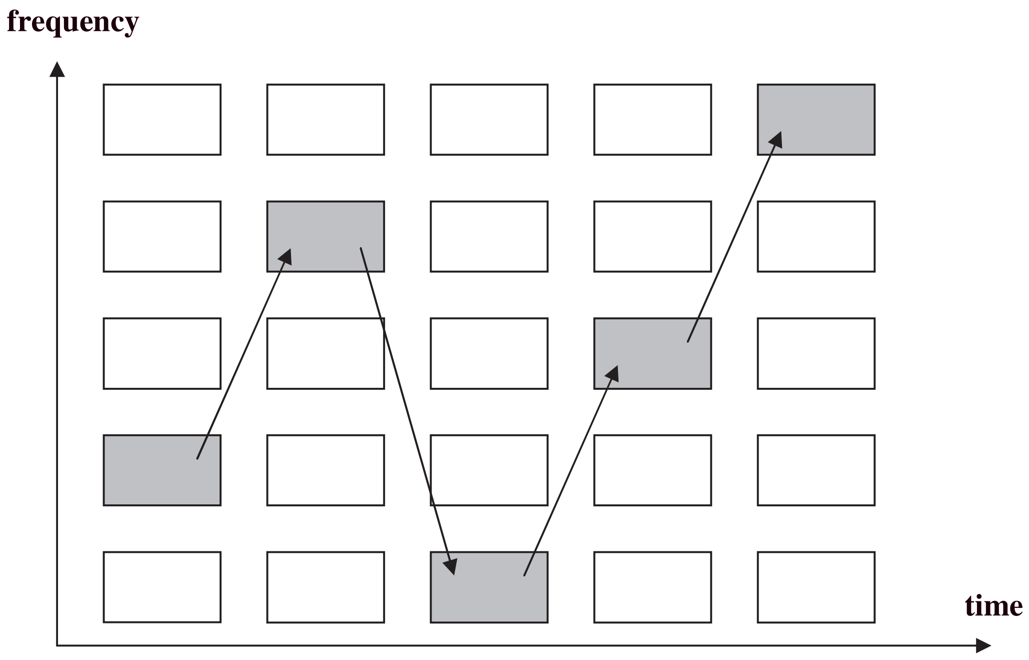


Figure 2: Frequency of a Bluetooth channel changing pseudo-randomly.

Authentication allows a user to limit connectivity to specified devices. Encryption makes data readable only to authorized devices/parties [1], hence preventing eavesdropping.

All Bluetooth-enabled devices implement the Generic Access Profile, which defines a security model that includes three security modes [1]:

1. Insecure mode. No security procedures are carried out.
2. Service-level enforced security. No security procedures are carried out before a channel is established.
3. Link-level enforced security. Security procedures are initiated before link setup is complete.

The in-built security measures of Bluetooth are important for this project, which deals with access control. However, these inherent security measures will not be sufficient for this application, as will be explained later.

2.7 Bluetooth Power Consumption and Operating Modes

Bluetooth supports four modes of activity in order to save power. In order of decreasing power consumption, the four modes are [6]:

1. Active mode
2. Sniff mode
3. Hold mode
4. Park mode

In this thesis, we are concerned mainly with the active mode and the hold modes, since they correspond mainly with the main procedures of the mobile phone (see section 6.2.2).

In active mode, the master and slave communicate with each other on the same channel. This would obviously use the most amount of power. This mode is used when there is data communications between both master and slaves. [6]

In hold mode, there is no active communications between master and slave. The slave merely listens to the channel, to see if it should exit this mode. During this time, the slave is able to scan, page or inquire about devices in the same area. This mode consumes significantly less power than the active mode, as noted in the following table. [6]

	Modes of Activity	
Product	Power Dissipation in Park Mode	Power Dissipation in Active Mode
CSR Bluecore 01	0.3 mW	0.6 – 135 mW

Table 2: Comparison of the differences in power dissipation in the two modes of activity. (Table taken and extracted from [6].)

2.8 Bluetooth Inquiry and Connection establishment

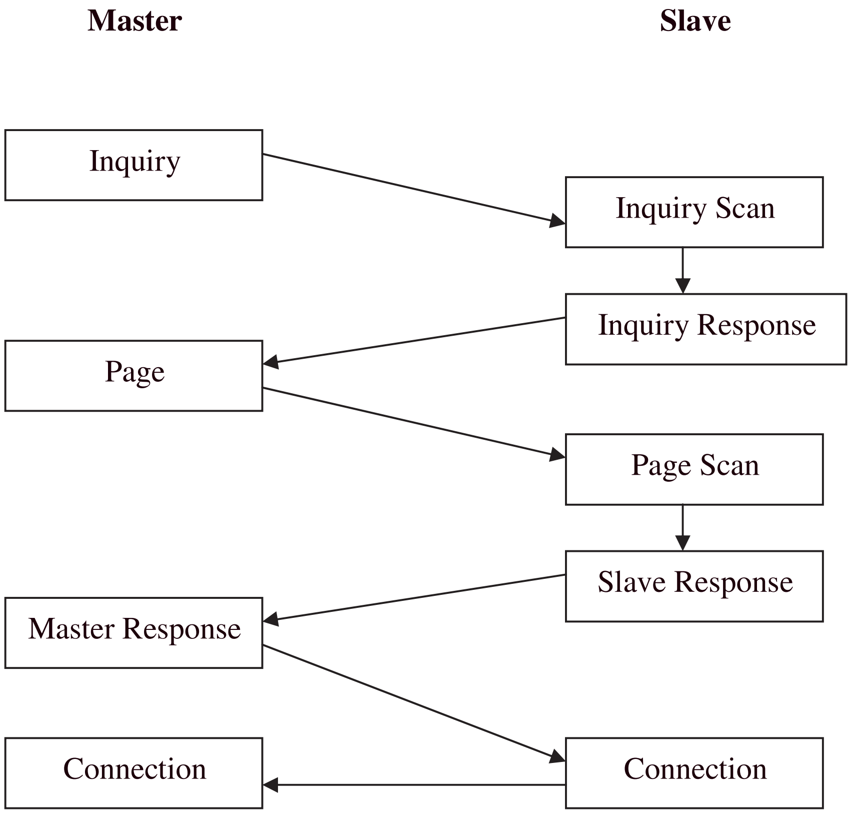


Figure 3: Typical Bluetooth connection procedure. (Taken from [7].)

The inquiry procedure is an important part of Bluetooth. It is a process where one Bluetooth device tries to find all neighbouring Bluetooth devices. A device that is trying to scan for other devices is said to be in ‘inquiry mode’. The device that listens for an inquiry request is in ‘inquiry scan mode’. This ‘inquiry scan mode’ is usually set in a Bluetooth device by setting it to be ‘discoverable’.

When inquiry is initiated, the device goes into ‘inquiry mode’ and accelerates its hopping frequency. On the other hand, the device in the ‘inquiry scan mode’ reduces its hopping frequency. This algorithm will allow the inquirer to catch up with the transmit frequency of devices that is in ‘inquiry scan mode’. This is important because of the frequency hopping algorithm employed in Bluetooth. When the frequencies coincide, the scanned device will act as slave and send its address and clock information to the master. [8]

After inquiry, the inquirer will be able to initiate connection to the inquired device. Since the connection is initiated by the inquirer, it will act as the master. This initial connection is called paging. Paging is done by the master by sending paging requests to possible slave frequency slots. This frequency slots are calculated from the Bluetooth address and the clock information received during inquiry.

At the time of connection establishment, the slave will synchronize its timing to that of the master's. Throughout the connection, the master never changes its hopping sequence or phase (current hop slot, determined by the master's clock). In contrast, the slave will have to synchronize with the master's clock all the time [7].

2.9 Bluetooth Software Stack

Bluetooth can be defined as a layered protocol architecture consisting of protocols such as the cable replacement protocol, the telephony protocol, the adopted protocol and the core protocols [1]. The layers are specified to add abstraction and to adapt the Bluetooth technology to other existing protocols.

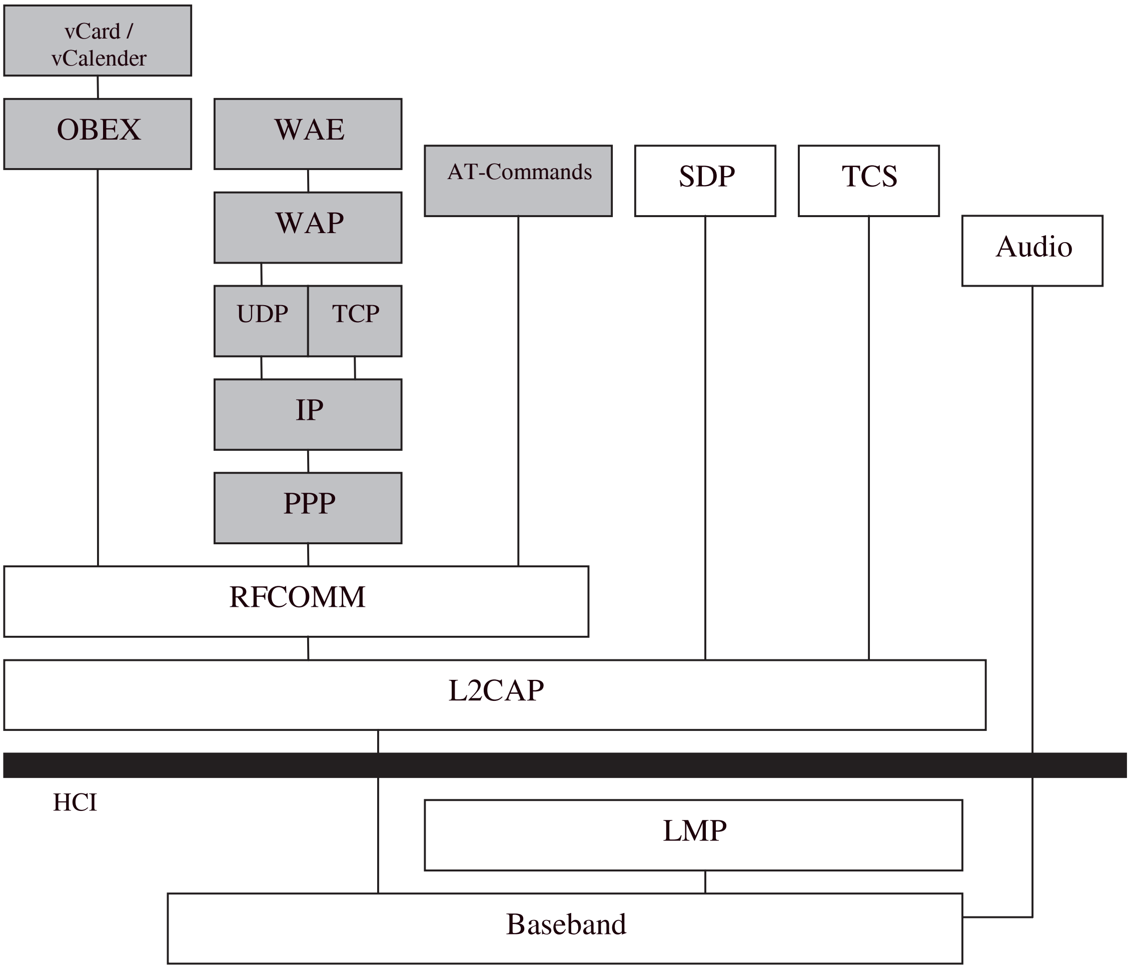


Figure 4: The Bluetooth Protocol Stack. (Diagram reproduced from [1].)

The core protocols are four protocol layers consisting of [1, 9]:

- **Baseband.** Specifies details of the air interface, including frequency, the use of frequency hopping, modulation scheme, and transmit power. Also specifies the protocol for connection establishment within a piconet, addressing, packet format, order of transmission, timing sequence, power control and channel coding.
- **Link manager protocol (LMP).** Responsible for link setup between Bluetooth devices and ongoing link management such as encryption and security. Allows service discovery, which detects other Bluetooth devices when in range.
- **Logical link control and adaptation protocol (L2CAP).** Adapts upper layer protocols to the baseband layer. It is also responsible for the multiplexing data to upper layer. This is done by assigning a particular PSM (multiplexer number).
- **Service discovery protocol (SDP).** Query devices of possible services available. Does not require a connection to be made.

We are interested in three particular layers, as they are the communication layers that we are likely to implement our system in. These layers are [1, 9, 10]:

- **HCI** (Host Controller Interface) sits between the L2CAP and LMP. It is the first interface between programmer and the protocol, allowing access to the hardware capabilities. It is also responsible for the multiplexing data to upper layer.
- **L2CAP** layer, as mentioned above acts as an adaptation layer to the upper layer, but it also allows connectionless data transfer.
- **RFCOMM** layer is a cable communication protocol designed to emulate serial ports.

We decided to program at the L2CAP layer because RFCOMM only provides one-to-one connections while L2CAP allows multiplexing of data via the PSM number. This means that two Bluetooth devices can have more than one connection to each other, allowing different Bluetooth programs to run simultaneously. This may not be important for our current implementation, but it may be useful for future implementations.

3. Bluetooth Applications

Most of today's Bluetooth applications can be grouped into several categories, namely

1. Office applications
2. Ad-hoc Networking
3. Access control applications

Office applications often relate to the setting up of a wireless office environment. For example, the keyboard and mouse of a computer are connected to the computer itself via a Bluetooth link, or connecting several computers to a Bluetooth-enabled printer. It also involves the synchronization of information between computers. For example, a business man who travels a lot would write a lot of information on his PDA. When he gets home from work, he will want this information to be added to his laptop computer. This can be done easily using if both his PDA and his laptop computer are Bluetooth-enabled.

Ad-hoc networking involves several devices in range, forming a community of networks (Personal Area Networks) that only exists as long as it is required. Current applications include the exchange of electronic business cards between users.

Bluetooth devices can be used in access control applications. One application is that it can be for mobile payment, like a credit card. A Bluetooth device such as a Bluetooth-enabled mobile phone could communicate with a Bluetooth-enabled cinema ticketing machine, for example, allowing the user to purchase a ticket without queuing, and then billing the appropriate amount to his phone bill. Alternatively, each Bluetooth device could contain a unique code which is used to identify a particular user, and how much credit he has in his credit account. Bluetooth can also be used as an access card. A user can access controlled areas of a building, or certain resources, by just being there, without having to swipe his/her card in a swipe machine.

4. Hardware and Development

4.1 Mobile Phone

The mobile phone would run the phone version of the ZoneIT software. The software would turn the ringing tone volume of the mobile phone down if the phone detects a nearby access point.

Minimum requirements for the mobile phone are:

1. Programmable.
2. Bluetooth-enabled.
3. API must have access to Bluetooth functionality.

To allow Bluetooth programming, a Bluetooth protocol (software) stack is required. This protocol stack is explained earlier in section 2.9, and it is usually implemented in the SDK. The SDK can be seen as simply a library that allows a programmer to control the Bluetooth devices.

Our team had to make a decision regarding the mobile phone we should consider using in the system. The following are the mobile phones which were considered.

4.1.1 Nokia 6310i

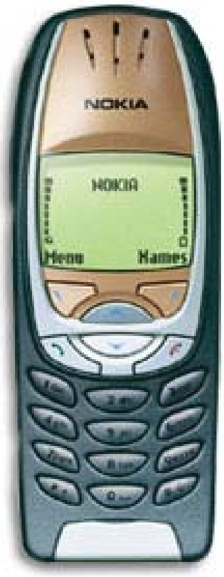


Figure 5: The Nokia 6310i.

This Bluetooth-enabled mobile phone was provided to our team by the University, through our supervisor, Dr Tim Moors. The initial plan was to program the phone using Java, which was the only way to program this phone, according to a Nokia representative we spoke to. However the Nokia representative also mentioned that the Bluetooth functionality of the functionality was not available to the programmer. Hence, it was not possible to use this mobile phone for our project.

4.1.2 Sony Ericsson P800

Figure 6: The Sony Ericsson P800.

This mobile phone was available to our team through Agus, who happened to own the phone. After doing some research on this phone, our team discovered that the phone ran on the Symbian Operating System, which provided an API for the Bluetooth functionality of the mobile phone. This made development on the mobile phone possible.

The Symbian Operating System is a popular operating system for mobile devices such as PDAs and mobile phones. This means that it would be easy to port our phone code over to these mobile devices easily.

4.2 Access Point

The access point is used to control the ringing volume of the surrounding mobile phones running the ZoneIT software. It acts as intelligent beacon, authenticating itself with the mobile phones using the ZoneIT software.

Minimum requirements for the access point are:

1. Bluetooth capability.
2. Programmable, supports Bluetooth discovery and data communications.
3. Ability to store and process Bluetooth addresses.

As with the mobile phone, a Bluetooth protocol stack is also required.

We have several choices to make regarding the platform to deploy the access point, and the operating system on which the access point would operate.

1. Bluetooth-enabled IPAQ PDA (Personal Digital Assistant).
2. Windows and Casira CSR BT device.
3. Windows and USB Dongle.
4. Linux and USB Dongle.

4.2.1 IPAQ and Microsoft Windows CE

The IPAQ PDA has many appeals. It is small and quite powerful, while running on the Windows CE operating system. The major disadvantage is that it cost a little over $1000 for a typical PDA, which implies that using the IPAQ as an access point would not be very cost effective when it comes to deployment.

Even so, our team was attracted to its simplicity and elegance. We decided to try developing the access point on the IPAQ. After much research, we found out that the SDK (software development kit) for the Pocket PC is freely distributed. Unfortunately, the SDK requires either Microsoft eMbedded Visual C++ development, or Microsoft eMbedded Visual Basic development to be used [11]. Those programming environments (IDE) will cost thousands of dollars, and hence we decided against using the IPAQ.

4.2.2 Microsoft Windows & Casira

Figure 7: Carisa micro-controller

The Casira micro-controller has a full Bluetooth protocol stack running on the BlueCore Chip. It is a combination of micro-controller and Bluetooth radio device called BlueCore Radio Module [12]. Using Casira is attractive because it can act as a stand-alone device. But its major disadvantage is that there is a limitation in the micro-controller's on-board memory. This could possibly impose a limit on the length of our code. Moreover there were also other students in the Bluetooth laboratory competing with us for the use of these modules. Hence we decided to go for other alternatives.

4.2.3 Microsoft Windows and USB Dongle

The Microsoft Windows platform coupled with a Bluetooth USB device (dongle) is another attractive combination to implement the system with. The university already has many computers running on Microsoft Windows in lecture theatres. By adding Bluetooth dongle and access control software, we can implement the system in our university lecture rooms. After research, we found that the SDK from Microsoft was free [13]. Unfortunately, after testing on two different computers we discovered that the Bluetooth stack in the SDK did not work properly, since we were unable to access the Bluetooth stack functions. This could possibly be because the Bluetooth USB dongle we were using was not supported. Similar remarks regarding this problem were also found on various forums over the internet.

4.2.4 Linux and USB Dongle

Similar to the previous case in section 4.2.3, this option is attractive because computers are widely available in the university. In addition, the Linux operating system is part of GNU, so it is freely available and downloadable from the internet. GNU software means that the public is allowed to distribute, use, and modify the software [14].

After some research, we found a Bluetooth stack for Linux called Bluez. At that time, Bluez had become the official Bluetooth stack for Linux. Just like Linux, Bluez is also GNU software.

4.2.5 The Chosen Platform and Operating System

After having evaluating and testing these options, and weighing the pros and cons of each, we finally decided to implement the access point program on Linux with a USB dongle, as detailed in section 4.2.4.

The advantages of using Linux with the Bluetooth USB device are:

- The cost for a Bluetooth dongle is relatively low (approximately $60).
- The Linux operating system and Bluez Bluetooth stack are free.
- Uses C programming language that comes with most Linux Operating System. Such as GCC or CC.
- Bluez implements most layers in the stack such as SDP, RFCOMM, L2CAP, HCI layers.
- It supports many Bluetooth devices, including the MSI dongle we are using [15].

We used Mandrake Linux Version 9.1 with Bluez. The installations for Bluez onto Linux will be described in the Appendix.

4.3 Programming Tools and SDKs

This section shall describe the programming tools and software development kits (SDKs) which were used in the development of our software system. Since both mobile phone and access point are different platforms, each has their own set of tools and SDKs. Also discussed are some of the advantages in using these tools and SDKs.

4.3.1 Mobile Phone

4.3.1.1 The Symbian Operating System, and its Bluetooth Stack

The Symbian operating system is the operating system on which the Sony Ericsson P800 runs on. The Symbian operating system comes in several variants. For example, the Series 60 mobile phones run on the Series 60 Symbian operating system, while the Sony Ericsson P800 runs on the UIQ Symbian operating system. All these operating systems are similar, and most of the code implemented on one Symbian operating system and can be ported rather easily to another Symbian operating system.

The Bluetooth stack provided by the Symbian operating system implements all the necessary components that define the Bluetooth specifications v1.1. In addition, it gives the programmer access to RFCOMM, L2CAP, SDP, and to a limited extent, HCI layers. Since our team is programming at the L2CAP layer, this Bluetooth stack is sufficient for our requirements. [16]

4.3.1.2 Programming on Symbian

Programming on Symbian was straightforward, except for the fact that the code had to be compiled and deployed on the mobile phone before any testing of the code could be done. Under such circumstances, an emulator for the phone would be ideal. Sony Ericsson does provide such an emulator, but at a hefty cost. However, even if we are willing to pay for the cost of the emulator, it would be practically impossible to test our Bluetooth code directly on the emulator, since the emulator does not recognise external Bluetooth devices.

Starting out on programming required the study of existing code. The code our team used was the “HelloWorld” example. Our team modified this code, and it formed the application framework of our software system. Since the focus of this project is Bluetooth development, we did not want to spend too much time on the user-interface.

Sony Ericsson provides an online-forum, which only provides technical assistance. Most of the time, it did not help. However there are several websites that have been established by Symbian programmers. Although most of these sites do not focus on UIQ Symbian programming, they were of much help because as mentioned before, most Symbian code can be ported easily across the variants of the Symbian operating systems.

Most of the information we required for the phone was on the documentation, which is packaged together with the Symbian SDK. The information was very useful and was critical to the success of the mobile phone software. However, there were some problems in the documentation. For example, there were errors in the example codes, and several functions were not well-described in the documentation.

4.3.2 Access Point

4.3.2.1 Linux and Bluez

Linux is a well known open source operating system released under the GPL (GNU Public License). It means that it is free for use or distribution. Recently, Bluez has been made the official Linux Bluetooth stack. Thus, a more recent Linux distribution with version 2.4.6 or higher would include Bluez packages. This also suggests that Bluez has been recognized as a reliable stack compared to other Linux Bluetooth stacks, and it will receive more attention and improvements by the open source community in the future. Bluez was originally developed by Qualcomm Incorporated.

Other well-known Bluetooth stacks are OpenBT, and Affix. Both are also open source. OpenBT has been somewhat static in terms of updates and improvements, while Affix is a newer stack that is developed by the mobile phone company Nokia.

The rationale for using Linux and Bluez was initially due to economic reasons, as other combinations of Bluetooth stack and operating systems are expensive. But as the project moved along, we discovered many positive aspects of Bluez.

Bluez implements almost all of the available stacks that is defined by the Bluetooth standard. The lists of implemented stacks taken from the Bluez official website are: [17]

- Bluetooth Core - HCI device and connection manager.
- HCI USB, UART, PCMCIA and VHCI (Virtual HCI) driver.
- L2CAP- Reliable datagram protocol.
- RFCOMM – Serial port emulation. Reliable connection oriented streaming protocol.
- BNEP - Ethernet Emulation.
- SCO – Synchronized connection (voice).

Bluez supports link layer security, and multiple connections. Those are crucial for our project. On top of those, it also allows multiple Bluetooth devices, which may be required for future extensions to this thesis. For example, we can use multiple devices create a single "virtual" access point. This enlarges the coverage area of the service.

4.3.2.2 Programming on Bluez

Bluez is an open source project [17]. Hence there is not much support given to programmers using the stack. The API (Application Programmer Interface) provides references to functions available to a programmer. Unfortunately, an API does not exist in Bluez. Hence, in order for a programmer to program using Bluez stack, they will have to look at the given example files, and also try screening the Bluez mailing list for any topics related to the development of the program.

The files given from the packages were in the form of C (source) files and H (header) files. There was little, or no documentation made regarding the functions used in those files. Thus it was considerably hard for us to start on the programming aspect for the access point. But the mailing list and the FAQ (Frequent Asked Questions) helped in the understanding of the source code, which provided the basis for our access points programs.

Fortunately for us, Bluez L2CAP is a socket based programming that uses standard UNIX C socket programming. Hence we were able to understand the basic requirement easily from other sources.

4.3.3.3 Portability

As this report mentioned earlier, Bluez uses Unix C, and it also uses standard Socket programming. This is mentioned in the Bluez web site that says, “Adding Bluez support to any existing socket based programs is very easy. For example, you would use AF_BLUETOOTH instead of AF_INET (ip) when you make a socket call. You would use sockaddr_l2 instead of sockaddr_in and SOCK_SEQ_PACKET instead of SOCK_STREAM and so on. Only a few new data structures and constants are introduced.” [17].

This means that it is easy to port from other C socket program to Bluetooth C socket. It also implies that it will be relatively easy to port a C socket for Bluez, to another stack or even another platform. After doing the programming for the access point, our team believes that the claims hold true for communication module, which uses L2CAP socket based programming. But the same cannot be said for the scanning module which uses HCI layer that is particular only to Bluetooth.

5. Thesis Proposal

5.1 Problem Statement

Currently, most access control solutions have been done via the use of conventional technologies, such as bar-coded swipe cards, and access pin numbers. With the introduction of newer mobile devices equipped with Bluetooth, it is possible to replace older access control technology with the use of Bluetooth technology. Bluetooth allows the development of wireless access control applications. Our thesis attempts to demonstrate the use of Bluetooth technology in access control applications. Our group has selected one particular application, which is described below.

The ringing tones of mobile phones often interrupt a meeting, such as a lecture, which is being carried out. This is often due to the mobile phone users forgetting to turn off their mobile phones prior to the meeting. This not only annoys the other parties in the meeting, but also interrupts the meeting that is taking place.

5.2 Proposed Solution

A practical solution to this problem would be to turn off these mobile phones automatically if possible, as they enter a designated “silent” zone, such as the meeting area. In that way, even if the owners of the mobile phones forget to turn them off manually, the mobile phone ringing tones will not interrupt the meeting when there is an incoming call.

This concept could then be extended to other access control applications such as using a Bluetooth-enabled phone to gain access to a room, which is secured by a Bluetooth-enabled lock. Another example would be to create an application that could provide a guided tour of a museum, for example, with text, audio and images being provided for the location the visitor is in.

5.3 Requirements

Our project team would like to propose, design and implement a system which solves the above-mentioned problem. The following gives a top-level description of how the system should behave:

- A user with a mobile phone enters a designated “silent” zone. The coverage area of an “access point” defines the zone.
- When the user enters the “silent” zone, the user’s mobile phone would be able to detect the presence of the access point.
- The mobile phone should be able to tell from the access point that it is indeed in a “silent” zone, and should turn down its ringer volume as required.
- The mobile phone keeps its ringing tone volume down as long as it is within the coverage area of the access point. If it exits that coverage area, the volume on the mobile phone will be restored to its previous state.
- The system would have to be resilient to two forms of denial-of-service attacks.
  - A malicious Bluetooth device should not be able to turn off the ringing tone of a mobile phone.
  - The access point should be robust, such that an attacker would not be able to bring down the services provided by it.

The requirements of the system are defined loosely so as to allow for different approaches to the problem. The next section shows the possible approaches which should meet the requirements.

5.4 Possible Approaches

The following are three possible approaches our team came out with during the formulation of a solution to the problem. Each solution has its own advantages and disadvantages.

5.4.1 Approach 1: Passive Scanning

There would be no actual data communications between the mobile phone and the access point. The Bluetooth addresses of the access points of the system would be either hard-coded into the mobile phone software, or accessible via a database in the mobile phone. When the mobile phone does a periodic inquiry process and detects nearby Bluetooth devices, it would compare the Bluetooth addresses of these devices with those in its database. If there is a match, the mobile phone would know that it is in a “silent” zone and would shut off its ringing tone.

The advantage of such a solution is that it is simple to implement, since we would not have to do any form of socket programming, and we would not have to do any form of programming on the access point. Another advantage is the scalability of the system, since each mobile phone is responsible for shutting itself down. The access point merely acts as a beacon with no processing requirements.

The disadvantage of such an approach is that there is no way to implement any form of application layer authentication, meaning that the system would be susceptible to denial-of-service attacks (see section 6.5.6). This may occur if a malicious user spoofs the Bluetooth address of a genuine access point, allowing him to illegally shut off the ringing volume of a mobile phone. Moreover the mobile phone would need to know before-hand the Bluetooth addresses of all access points, meaning that the databases on the mobile phone may need to be updated frequently. This makes the deployment and maintenance of such a system difficult.

5.4.2 Approach 2: Data Communications and Passive Scanning

The mobile phone would run a server and the access point would run a client. The access point would periodically scan for nearby Bluetooth devices. Once it finds a Bluetooth device (phone), it attempts to set up a communication channel (via a socket) with the phone, and authenticates with it. Once authentication is complete, the phone shuts off its ringing tone and records the Bluetooth address of the access point. It then continues to inquiry for nearby Bluetooth devices, checking if the access point is still in the coverage area using the recorded Bluetooth address.

The advantage of this approach is that it is possible to authenticate the access point, regardless of its Bluetooth address. Hence there is no need for hard-coding the Bluetooth address of all possible access points or for storing all those addresses in a database. This approach also enjoys scalability. The access point only needs to authenticate each mobile phone once, and ignore it for the rest of the time. The mobile phone would determine if it is still in the coverage area of the access point.

The disadvantage is that it is still possible to spoof the Bluetooth address of the access point, allowing a malicious user to fool a phone using the software into thinking that it is still within the coverage area of an access point, when it is actually not, in the inquiry phase (i.e. after the authentication phase).

5.4.3 Approach 3: Regular Polling

This approach is similar to second approach (5.4.2). However, the access point would attempt to establish a communication channel with the phone at every predefined period (requires a timer), instead of just once. The phone would not have to do any inquiry of nearby Bluetooth devices. As long as the access point manages to communicate with the phone at that time interval, the phone ringing tone remains turned off. Otherwise, the phone would assume that it is outside the “silent” zone and restore its ringing tone.

The advantage of this approach is that it is more secure than the other two approaches. This is because the access point is being authenticated at regular intervals. A malicious user may still spoof the Bluetooth address of the access point, but would not be able to authenticate itself with the mobile phone. Moreover, in such an approach, all the processing load is transferred from the mobile phone to the access point. Since a mobile phone has significantly less processing power than the access point (which is powered by a PC), this may be an important factor worth considering.

One significant disadvantage of such a approach is the scalability of the system. If too many users attempt to use the system simultaneously, the access point may have trouble coping with these users as it tries to authenticate all of them at regular time intervals. It is possible that the access point may be in the midst of authenticating a list of users, when the next list of users arrives, leading to an accumulation of backlog of users it has to authenticate. Another disadvantage is the energy requirements. Such regular polling will require regular data communications (active mode), and this will consume a lot of battery power from the mobile phone, as described in section 2.7.

5.5 Choice of Approach

Our team has decided to use the second approach, as described in section 5.4.2. This is because of the possibility of our system being deployed in a situation where there are lots of users, such as in a lecture theatre. This means that the system has to scale to a large number of users, and hence the third approach described in section 5.4.3 cannot be taken. Also, because we want our system to be secure, we cannot use the approach as described in section 5.4.1. Although the second approach is susceptible to spoofing, our team feels that it is a good compromise between approach 1 and 3, and hence we have decided to take this approach.

It is possible to allow the user to be able to select between the three approaches in the mobile phone software system, since these three approaches can be programmed as independent components of the system. However our team has decided not to do so, due to the lack of time, and have left it as a possible extension for future work.

5.6 Specifications

From the above requirements and approach, our team has developed a set of specifications which our system would have to adhere to, in order to provide the required functionality.

5.6.1 Mobile Phone / Access Point Interface

- Both the mobile phone and access point would communicate via the Bluetooth v1.1 specifications.
- Using the client-server relationship model of most communication systems, the mobile phone would act as a server, while the access point would act as a client. The access point would access the mobile phone.

5.6.2 Mobile Phone

- Upon starting up, the software shall run a server module, which listens for incoming Bluetooth connections.
- If an incoming connection is initiated, the mobile phone would have to ensure that the owner of the incoming connection belongs to a trusted access point. If not, it would close the connection and continue listening for new connections.
- After authentication, the mobile phone deduces that it is in a “silent” zone, and automatically turns down its ringing tone volume. The Bluetooth connection is then closed.
- During this period, the mobile phone does a Bluetooth “inquiry” on the access point periodically, in order to ensure that it is still within the coverage area of that access point.

- Because Bluetooth is a wireless communications standard, and it experiences frequent dropouts, this “inquiry” procedure would have to fail several times before the mobile phone decides that it is not in the coverage area of the access point. If it fails too many times, the software stops the “inquiry” procedure, and restores the volume of the ringing tone.
- Once out of the coverage zone, the software restarts the server module to listen for new incoming connections.

5.6.3 Access Point

- The access point scans for all nearby Bluetooth devices inside its coverage area. The access point will keep a list of all the devices in the form of Bluetooth addresses, which is unique in all Bluetooth devices.
- The access point will then try to communicate with all the mobile devices, and disable the ringing tones of those mobile phones which have the ZoneIT software installed.
- During communications, the access point will be required to authenticate itself with the mobile device. This authentication is used to make sure that the access point is authorised to disable the ringing tone of the mobile phone.
- The access point must be efficient, possibly connecting to several mobile phones simultaneously.
- Finally, the access point should not reconnect to those mobile phones it has connected to before, since it would be redundant to do so.

6. System Design

6.1 System Design Considerations

Before starting on the design procedure, we need to enhance our basic understanding of Bluetooth communications, so that we can make the right decisions when carrying out the design.

The first issue is the connection initiation. Connections can be initiated either by the access point or the mobile device. After careful consideration, our team believes that if the mobile phone were to initiate the connection, they would have to screen through all the available devices in the vicinity. This is power consuming for the phones, and we do not want to burden these mobile devices in such a manner. Moreover, making the phones initiate the connections means that the access point will be a slave in the piconet. This will require the access point to attempt to track the frequency hops of only one of mobile device, but not the others. The consequence of this is that the access point can make at most one connection at a time.

In addition, it is more practical if the access point coordinates the connections, because if every mobile phone tries to initiate a connection to the access point, the access point will be overwhelmed by a large number of connections. This causes a scalability issue. Hence it was decided that the access point would initiate the connections, while being the master, during all communication and thus it will be the device that inquires (scans) all other surrounding devices, while the mobile phone listens for incoming access point connection requests.

The second consideration is that as the state of the device cannot be in connection and inquiry mode at the same time. This means that the device cannot scan for other devices while handling a communication. This implies that they have to be done consecutively. This is reflected in our design later on, when that part of the code is executed sequentially.

6.2 Mobile Phone

6.2.1 Software Modules

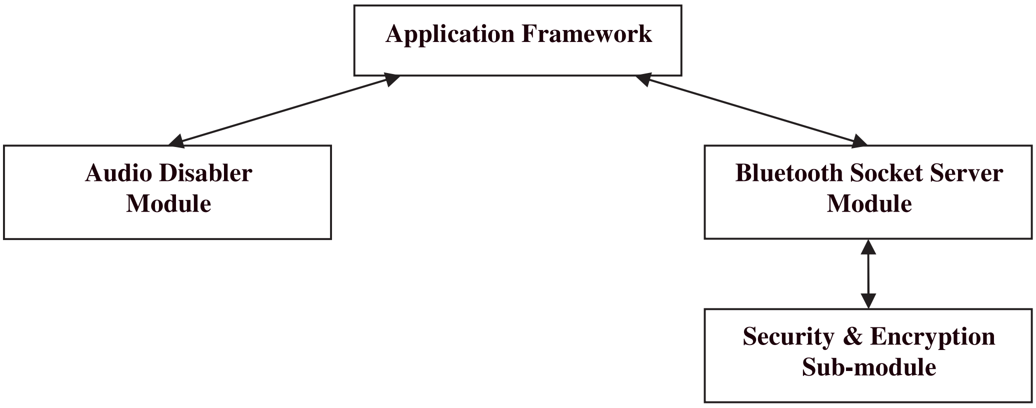

Application Framework
Audio Disabler Module
Bluetooth Socket Server Module
Security & Encryption Sub-module

Figure 8: Software modules of mobile phone, and their independencies.

In order to simplify the problem, we broke the program down into several modules. Each module can be designed and implemented independently. Finally they can be integrated.

The mobile phone software consists of three main modules, and one sub-module.

The ***Application Framework*** is the driver of the software. It consists of methods to construct the graphical user interface (GUI). From there, the other modules can be easily added, and the GUI modified to incorporate the additional functionalities these modules add. The application framework also provides an interface for communications between each module. If, for example, the *Bluetooth Server Socket Module* needs to access a method in the *Audio Disabler*, it will call it via this framework.

The ***Audio Disabler Module*** provides a simple interface to the Beatnik audio libraries (see section 7.1.2). It simplifies the audio disabling procedures by hiding these procedures from the calling application framework.

The ***Bluetooth Socket Server Module*** runs a Bluetooth socket server process in the software. This is the module that allows the mobile phone to exchange data with the access point. The data consists of mainly authentication related data, which is processed by the *Security & Encryption Sub-module*. After authentication, this module continues to inquire the access point. This allows the mobile phone to know if it is still within the coverage area of the access point. As long as it is, this module will continue running. Once outside the coverage area, this module is restarted.

6.2.2 Top-Level State Diagram

A state diagram shows the various possible states of the program during operation, and how the program changes from one state to another. It is a tool to design the flow of a software system. Figure 9 shows a state diagram of the Bluetooth Server Socket module.

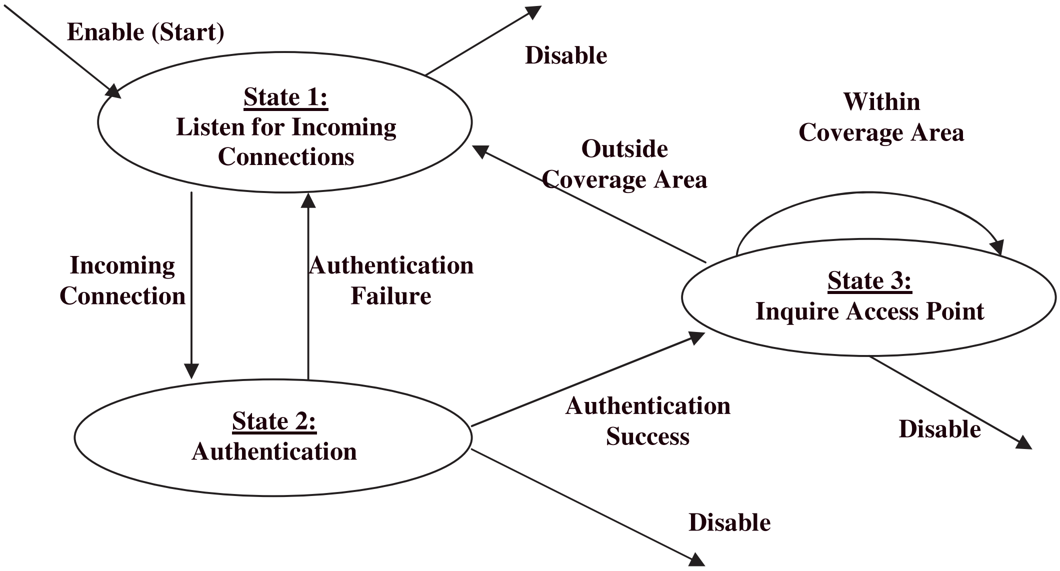


Figure 9: State diagram of the Bluetooth Server Socket module.

1. The ZoneIT software is enabled when the software is first run. This allows the software to listen for incoming Bluetooth connections to the phone (State 1).
2. If there is an incoming connection, the system moves on to the authentication phase (State 2), which is detailed in section 6.4. If authentication fails, the state machine is reset back to state 1, and the software continues to listen for new connections.
3. If authentication is successful, the state machine moves to state 3, where it will periodically inquiry for the access point, to determine if it is still within the coverage area of the access point. If it is, the state machine will remain in state 3. If not, the state machine is reset back to state 1, whereby it listens for a new connection.

4. The state machine can be terminated at any time if the user “disables” the software by shutting it down.

6.2.3 Inquiry State Diagram

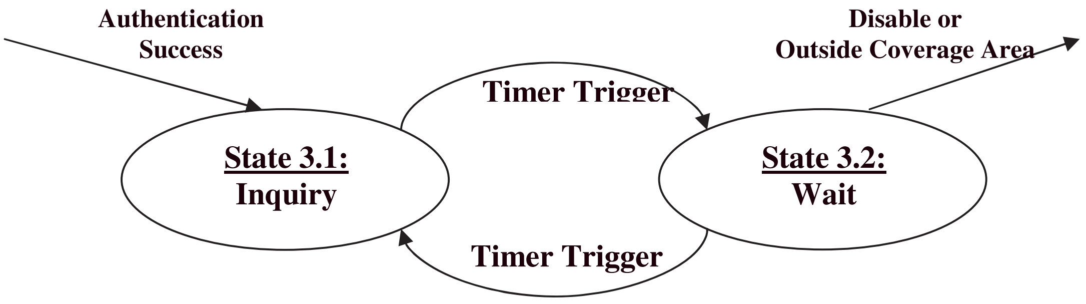


Figure 10: State diagram illustrating how the inquiry procedure is carried out.

- A timer trigger switches the state machine between the Inquiry and Wait states. In the Inquiry state, the software inquires the access point. In the Wait state, the software waits for a short moment, allowing other processes on the mobile phone to be carried out, so that the software does not hog all the resources of the system.

6.3 Access Point

6.3.1 Software Modules

The design of the access point starts with simple diagrams of how it interacts with other Bluetooth devices, and defining how it supposed to behave to satisfy the requirement. Then we proceed to define the required modules for the access point to function as intended. The diagram below shows the basic communication behaviour. For example, access point attempting to connect to devices which are not mobile phones will result in a "connection refused" error.

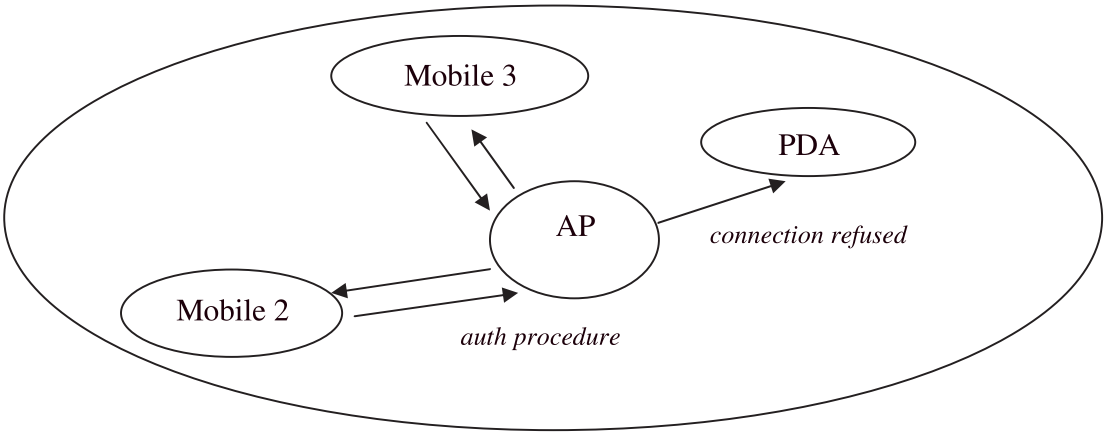


Figure 11: Relationship between the access point and mobile devices.

To reduce complexity and increase reusability, we introduce abstraction by use of modular design. That is we break the main problem into many smaller problems, and having each modules solving different problems.

As the design matures, it emphasizes more on security and efficiency, because we realize that the two factors are an essential aspect of any access control application.

The access point program is broken into the following modules:

1. ***Directory module*** acts as the record keeper of which phones has been deactivated, and when it was last seen.

2. ***Scanning module*** will discover surrounding Bluetooth devices.

3. ***Communication module*** provides Bluetooth communication to mobile phones.

4. ***Security module*** will encrypt and decrypt messages.

5. ***Main module*** will use all the other modules working together to provide the access point. This can be seen as the driver of the program controlling all the other modules in terms of synchronization of data, and program flow.

6.3.2 Data Flow Diagram

The modules interact by sending and receiving data from one module to other modules. A data flow diagram (DFD) will illustrate how the modules interact with one another.

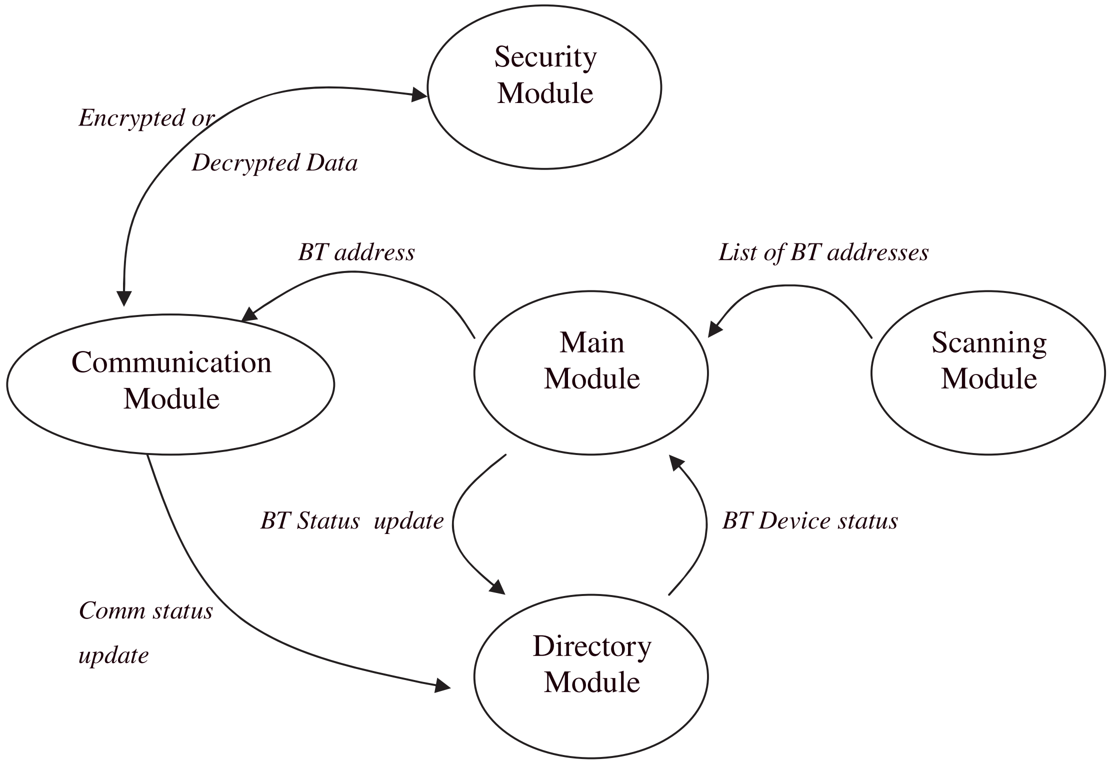


Figure 12: Data Flow Diagram of the access point modules.

The Scanning module and the Security module are dependant only on one other module. Hence, there is not much efficiency gained in terms of reusability, but it is desirable to introduce abstraction when coding for reasons mentioned earlier.

6.3.3 State Diagram

The following state diagram illustrates the various possible states the access point can have.

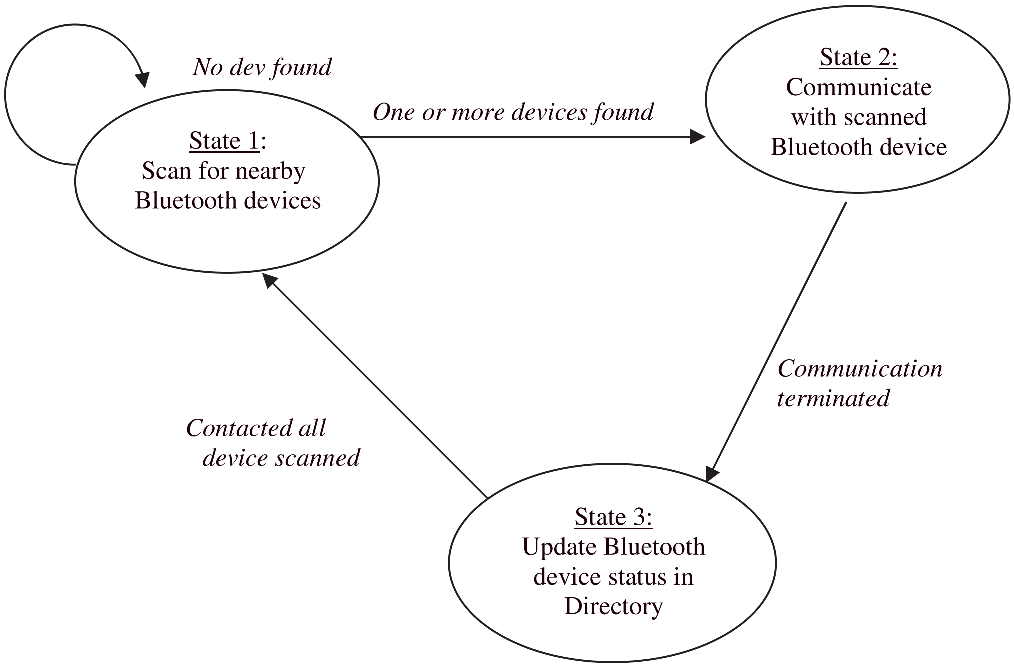


Figure 13: State diagram of the Access Point.

The diagram above only describes the high level states of the main module for the access point. In lower level, state 2 becomes more complex as parallelism is introduced by multithreading. Multi-threading will be discussed more in section 7.2.2.1.

The three states above can be seen as the lower layer of the main module, where the states are the procedure that the main module has to execute.

6.4 Authentication Protocol

The authentication protocol was designed to make the communications between the mobile phone and access point as secure as possible. The following diagram shows how a "nonce", in a form of a random number, is used together with an encryption algorithm to secure the communications. The introduction of the "nonce" in the communications makes a playback extremely hard, if a sufficiently large "nonce" is employed.

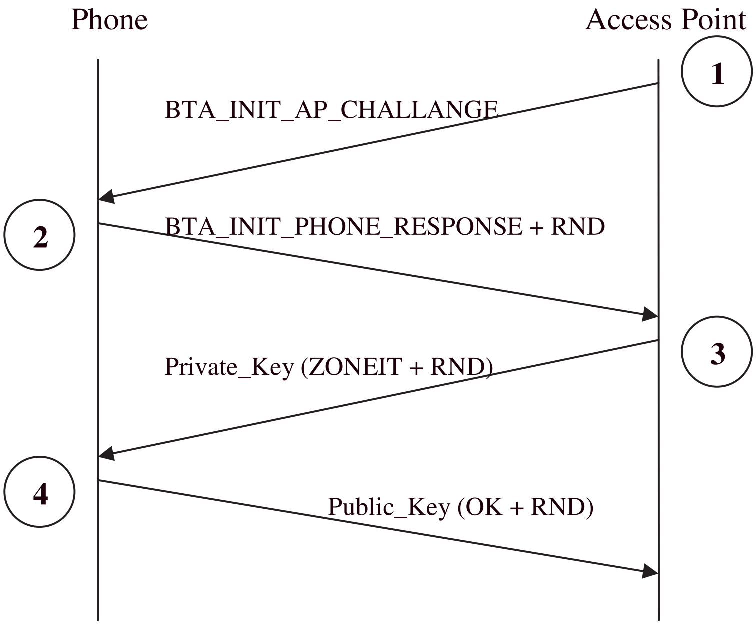


Figure 14: The figure shows the messages exchanged between the phone and the access point after a connection has been established between the two.

After the access point has initiated a connection to the phone, the following procedures are carried out in order for the mobile phone to authenticate the access point.

1. The access point issues a challenge (*BTA_INIT_AP_CHALLANGE*) to the mobile phone.
2. The mobile phone verifies the challenge. It generates an 8-bit random number (*RND*), and appends it to its response (*BTA_INIT_PHONE_RESPONSE*) to the access point, before sending the message to the access point. (See section 6.5.7 for more information about this random number.)
3. The access point will then request the mobile phone to turn off its ringing tone by appending the same random number (*RND*) to its request (*ZONEIT*), and then encrypting the whole message with its private key, before sending the message to the mobile phone. The encryption algorithm used is RSA, as described in section 6.5.2.
4. The mobile phone, on receiving the encrypted request, decrypts the request with the access point's public key. If the request is successfully decrypted, and the random number is correct, the phone then shuts off its ringing tone. It then generates an acknowledgement (*OK*) with the random number (*RND*) appended at the end, and encrypts this entire message with the access point's public key, and sends it back to the access point.

After these procedures, the mobile phone's ringing tone would be shut off, and the access point would note that that specific mobile phone has had its ringing tone shut off.

6.5 Security and Encryption

6.5.1 Public Key and Shared Key Cryptography

Public key cryptography is one whereby a user generates two pairs of keys for himself. One of these keys is termed a “public key”, which is made known to the public. The other is termed “private key” which the user keeps secret. These two keys are functional inverses of each other, meaning that if you use one to encrypt information, you will need the other to decrypt it. If public key cryptography is used in this system, the access point will possess the private key, while the phones will possess the public key.

Shared-key cryptography is generally used between two trusted parties only. These two parties share the same encryption key, which is also used as a decryption key. There are only two approaches to using this system in our system.

1. The access point only has one key. Everyone shares the same shared-key.
2. The access point stores a shared key for each user.

The first approach makes the system insecure. Since everyone knows what the encrypted key is, an attacker can simply use that key to encrypt and decrypt data moving across the network, allowing him to perform a denial-of-service attack easily.

The second approach makes the system not scalable to a large number of users. As the number of users in the system increase, the number of shared keys will increase. Moreover, introducing a new user to the system would require some form of “registration”, whereby the user is issued a new shared key. This makes the deployment of such a system cumbersome.

On the other hand, a public key cryptography solution will allow the public key of the access point to be hard-coded into the phone software, thus avoiding the scalability issue. The security issue is also avoided, since a malicious user only knows the public key of the access point.

6.5.2 Introduction to RSA

RSA is a public key encryption algorithm. The key length of the RSA algorithm is variable. Using a long key will provide a higher level of security, but requires more computation. Likewise, using a short key provides less security, but improves the efficiency of the algorithm. The most commonly used key length is 512 bits. [18]

There is no way to prove that RSA is secure. However, there is no evidence that anyone has managed to break the algorithm yet. RSA works on the fact that it is hard to factor a large number. Using the best known technique today to crack a 512-bit number would take literally thousands of years to complete. [18]

The reasons why our team chose RSA over other public-key algorithms were because RSA has been well-tested, and the source code is readily available (ease of implementation).

6.5.3 RSA Key Generation

The following steps are used to generate a private and public key pair [18].

1. Generate two large prime numbers, p and q.
2. Let n = pq
3. Let φ = (p-1)(q-1)
4. Choose a small number, e, which is relatively prime to φ.
5. Find d, such that de % φ = 1
6. The public key is (e, n). The private key is (d, n).

Note that the operator '%' in 'x % y' stands for 'the remainder of x divided by y'.

Both p and q should remain secret. Since p and q are large primes, the result key values will be large, and very hard to factorise.

6.5.4 Encryption and Decryption

The encryption and decryption of a message occurs as follows [18].

1. Let m be the message (plain-text), and m < n
2. The encrypted message (cipher-text) will be

$$c = m^e \% n$$

3. The decrypted message (plain-text) will be

$$m = c^d \% n$$

6.5.5 Using RSA in ZoneIT

This software system uses the RSA public encryption algorithm to encrypt its communications. However, implementing an algorithm with a large key strength is complicated. Available source codes on the web could not be ported over to the Symbian OS. Hence, we have settled for an RSA algorithm with a weaker key strength of 32-bits. Our focus in this project is not to implement the RSA algorithm, and hence a 32-bits algorithm would suffice to demonstrate our software system. Due to the modular structure of our design, switching to a stronger RSA key-strength would simply be a matter of replacing the encryption modules of the system.

6.5.6 Protecting the System - Denial-of-Service Attacks

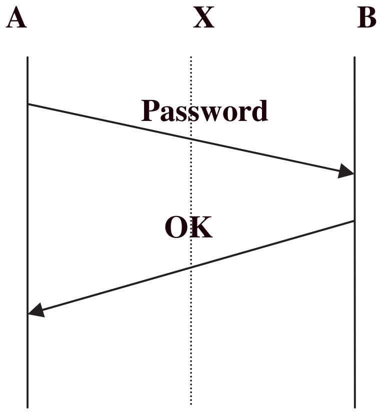
A denial-of-service attack is a type of attack on the services of a resource, causing a user or organization to be deprived of the services of that resource they would normally expect to have [19].

In the case of our project, there are two services. One of the services is the ringing tone of the mobile phone. A malicious user can deny the user access to his ringing tone by using a Bluetooth device to imitate the functionality of an access point, forcing users of the phone software to shut off their ringing tones unnecessarily. Unfortunately, encryption the communications between the mobile phone and the access point is insufficient to defend against such an attack, because of the possibility of a playback attack, as detailed in the next section.

The other service is the service provided by the access point. The access point has to be robust, and resist attacks to take its service down. However this robustness will not rely on the authentication procedure but rather on the implementation of the access point itself, and hence will not be discussed under this section.

6.5.7 Protecting the System - Playback Attacks

A playback attack is one in which a malicious user monitors and records the traffic of a user, and then later "plays back" the same activity, possibly trying to trick the system into believing that the attacker is a valid user.



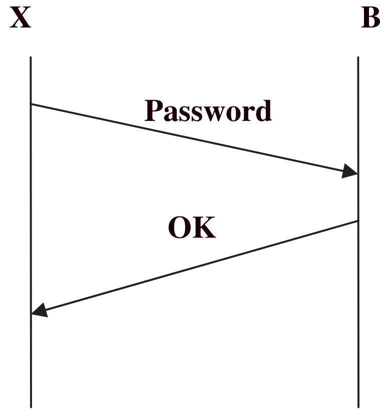


Figure 15: Shows an attacker X listening to the traffic between users A and B.

Figure 16: Shows the attacker X pretending to be user A by imitating the traffic sent from A at a later time.

As can be seen from figures 15 and 16, encrypting the messages between users A and B would not work, because the attacker X can just playback the encrypted messages and still be authenticated.

The only solution to this problem is to add some form of timestamp to the communications between A and B. For example, user A can append a timestamp at the end of the password, and then encrypt the message. User B, on receiving the encrypted message, can decrypt it and examine the timestamp. If the timestamp does not reflect the current time, then user B will know that the message originated from an attacker who is carrying out a playback attack. Because attacker X has no idea how to encrypt and decrypt the message, there is no way he can ever carry out a playback attack on the system.

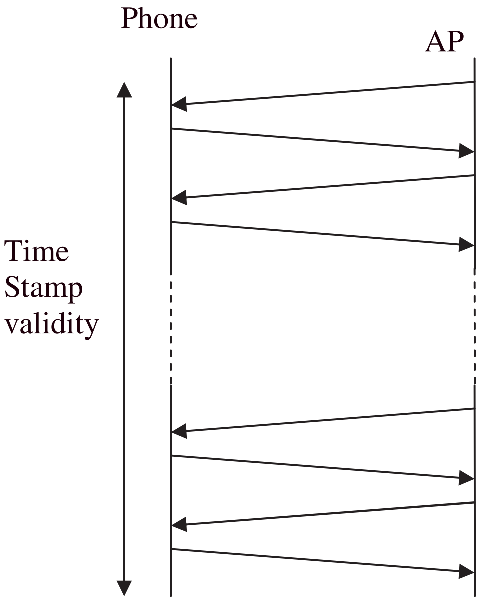
In this system, the timestamp can take two forms.

1. System time
2. Session ID / “nonce”.

Using the system time to create a timestamp is not difficult. However there is the issue of synchronisation of times between both parties. The success of such a timestamp would depend on the expiry time of the timestamp. If the timestamp is programmed to last for too long a period, a malicious user can take advantage by performing a denial-of-service attack before the timestamp expires. If the timestamp is programmed to last for too short a period, non-malicious traffic might be assumed to be malicious, and the authentication procedure cut short. Figures 16 and 17 illustrate the short-coming of using such a timestamp. The complexity of determining the expiry of the timestamp made our team decide against using such a timestamp.

The other alternative is to use a “nonce”, which is a random number. Provided the random number is sufficiently large, it would be hard for an attacker to attack the system. Moreover, we would not be faced with the difficulties of determining an expiry time for the timestamp.

Our system merely uses an 8-bit “nonce”, which is insufficient to defend against an attack, since an attacker can easily make a “guess” of the random number issued by the mobile phone. However, 8-bits are sufficient for demonstrating this software system. It would also not be difficult to add more bits to the “nonce” at a future date.



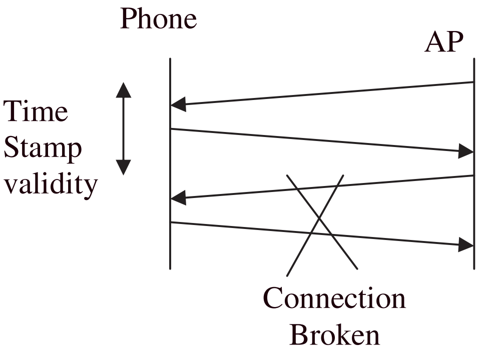


Figure 17: Time stamp expiring beyond the duration of a session. An attacker can use this excessive time to establish another session.

Figure 18: Time stamp expiring before the end of a session. Authentication will be forced to end pre-maturely.