NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY, MATHEMATICS AND ELECTRICAL ENGINEERING

NTNU

Security Specification, Access Control and Dynamic Routing for Ad-Hoc Wireless Networks applied to Medical Emergencies

Tor Inge Skaar Tor-Erik Thorjussen

SINTEF

NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET
FAKULTET FOR INFORMASJONSTEKNOLOGI, MATEMATIKK OG ELEKTROTEKNIKK

HOVEDOPPGAVE

Kandidatenes navn:	Tor Inge Skaar, Tor-Erik Thorjussen
Fag:	Telematikk
Oppgavens tittel (norsk):	**Sikkerhetsspesifikasjon, aksesskontroll og dynamisk ruting for ad-hoc trådløse nettverk anvendt i akutt-situasjoner.**
Oppgavens tittel (engelsk):	**Security Specification, Access Control and Dynamic Routing for Ad-Hoc Wireless Networks applied to Medical Emergencies.**
Oppgavens tekst:	The objective of this project is to analyze, define and implement selected security requirements in a medical emergency scenario. This is a scenario where a large number of patients and many and varying types of rescue personnel – with different roles – might be involved, from trained ambulance personnel and specialist physicians to semi-trained volunteers from e.g. Red Cross. The project shall base its work on the FieldCare prototype developed at SINTEF Telecom and Informatics.
Oppgaven gitt:	20. januar 2003
Besvarelsen leveres innen:	16. juni 2003
Besvarelsen levert:	16. juni 2003
Utført ved:	Institutt for Telematikk
Veileder:	Ingrid Svagård (SINTEF Telecom and Informatics)

Trondheim, 16. juni 2003

Stig Frode Mjølsnes
Faglærer

EXECUTIVE SUMMARY

There exists a manifold of potential security threats to a wireless network system. We examine many of these and categorize and analyze them in accordance with commonly recognized standards from ISO/IEC (International Organization for Standardization / International Electrotechnical Commission). Our scenario is based on the FieldCare project, currently under development at SINTEF Telecom and Informatics.

FieldCare is a system that uses wireless mobile devices together with medium/long range radio links and smart software for the gathering of information and the distribution of this information between all the people involved in dealing with a medical emergency.

This scenario is processing sensitive information; therefore we review and analyze the laws and regulations governing the proper and secure handling of this type of data. The Norwegian regulatory agency responsible for the enforcement of these laws and regulations, the Data Inspectorate, are contacted in order to get a verification of our interpretations.

Another aspect of this thesis is to study different methods for access control to be used on the mobile devices on the scene of the accident. Several well-known access control schemes are considered, and some of these are added to our prototype implementation. As for the access control specific to the login procedure, we present a solution that uses the iButton technology together with security mechanisms such as hash algorithms.

In addition, SINTEF Telecom and Informatics are interested in improving the current static network topology of the FieldCare system, so we suggest a solution that provides a better management, scalability and reliability of the ad-hoc wireless network on the site of the accident. Our solution utilizes a dynamic routing protocol called Optimized Link State Routing (OLSR), and the performance of this protocol is tested.

Our implementation are based on SINTEF's FieldCare prototype, to which we add extended functionality, including dynamic routing, access control with iButtons and the possibility of securely downloading remotely stored patient journals to the accident site. In addition, we replace the current use of the commercial SavaJe operating system on the mobile devices with the open source alternative; Linux. In this context we examine the difference between using open source software and closed source software in relation to the security provided by each alternative. The implementation of a new prototype is carried out in a close cooperation with another group, who also deal with the FieldCare system, in order to deliver a more coherent end result.

The most important future recommendations we suggest, are further analysis of the specific threats to the FieldCare system and the possible use of the Secure Ad hoc On-Demand Distance Vector Routing protocol instead of the OLSR protocol.

PREFACE

This report is the result of our master thesis, completed in the 10th semester at the Department of Telematics (ITEM), section for Information Security (IS), at the Faculty of Information Technology, Mathematics and Electrical Engineering (IME) at NTNU.

We would like to give thanks to our supervisor, Ingrid Svagård at SINTEF Telecom and Informatics. She has been very helpful and supportive with information and advices during this project. And we would also give thanks to our professor, Stig Frode Mjølsnes, for being helpful with expanding the original project description to allow more students to participate.

Computer equipment and other devices were borrowed from ITEM, and Jarle Kotsbak, Asbjørn Karstensen and Pål Sturla Sæther at ITEM deserves our gratitude for always being helpful when we needed equipment and assistance.

Many thanks also go to the other group working with the FieldCare project. It's members; David Henriksen, Ingar Melby and Einar Skjellerudsveen have all participated in a close collaboration with us to achieve the best possible end result.

Trondheim, June 16, 2003.

Tor Inge Skaar

Tor-Erik Thorjussen

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

This chapter contains general information about this thesis, and some practical information targeted to the reader of this report. The assignment as presented in 1.1 is the final version approved by both SINTEF and our professor, Stig Frode Mjølsnes.

1.1 ASSIGNMENT

The objective of this thesis is to analyze, define and implement selected security requirements in a medical emergency scenario. This is a scenario where a large number of patients and many and varying types of rescue personnel, with different roles, might be involved; from trained ambulance personnel and specialist physicians to semi-trained volunteers from e.g. Red Cross. This thesis shall base its work on the FieldCare prototype developed at SINTEF Telecom and Informatics.

1.1.1 SCOPE OF WORK

To study, suggest and implement security solutions for the use of personal electronic information tags, mobile terminals and ad-hoc Wireless Local Area Network (WLAN) in a medical emergency scenario.

1.1.2 ISSUES TO ADDRESS

Integrity, confidentiality, access control, scalability, privacy regulations, and “blue-light” (emergency) access.

1.1.3 SPECIFIC TASKS

1) To study the security issues and develop a security requirements specification.

2) To experiment with alternative open source operating systems underlying the FieldCare application on all types of devices, e.g. Linux, and analyze the security issues when using open source software (FieldCare is currently using the commercial SavaJe Operating System (OS)).

3) To study methods for access control and implement a solution for access control in the WLAN on site, e.g. using password-protected iButtons as personal tokens.

4) To implement a solution for dynamic routing in the WLAN. The FieldCare prototype uses currently a fixed peer-to-peer configuration.

1.2 PARALLEL ASSIGNMENT

The original thesis proposal was only meant for a few students. But two groups of in all five persons showed interest in the thesis. SINTEF Telecom and Informatics together with our professor, Stig F. Mjølsnes, expanded the thesis and divided it into two parts, though quite interconnected.

The first part, which is our project, was described in 1.1.

The other part is done by another group, which we will later simply refer to as "Group 2". Their task is to analyze and develop a security requirement specification in a medical emergency scenario, and implement selected security requirements. The project shall base its work on the FieldCare prototype developed by SINTEF Telecom and Informatics. After identifying a patient, the medical personnel may want to download parts of the patient's electronic journal to the mobile devices used on the scene of the accident. They will also analyze, recommend and implement a security solution for the communication link between the mobile device and the hospital server [HMS03].

Figure 2.6 in section 2.5 shows graphically what we and Group 2 will focus our work on, in relation to the whole system.

1.3 READER'S GUIDE

In this report, we have used "`Courier New`" to indicate the following:

- Content of files
- File names
- Directory names
- Script names
- Commands
- IP (Internet Protocol) Addresses
- System input and output
- Program variables

Italic was used to emphasize certain words, phrases and functions.

References to external sources are marked with square brackets [ ] and more information about each reference can be found on page 94.

Tables and figures are labeled with the chapter number together with a sequence number. As an example, figure number 26 in chapter six will be labeled Figure 6.26.

To be able to get the most out of this report, the reader should have a technical education on a university level and knowledge in information security, together with a general knowledge of Linux and Java.

1.3.1 STRUCTURE OF THE REPORT

The general structure of this report follows the order of the specific tasks from 1.1.3, with the exception of the task about experimenting with other operating systems.

The report starts with giving an overview of the FieldCare project which this thesis is based on. You will here be given some background information about FieldCare, a description of the FieldCare scenario, a presentation of our ideas and definitions of various terms used in FieldCare.

In chapter 3 we will analyze the security in the FieldCare scenario. We will categorize and analyze different threats and also examine some security and privacy regulations.

We will then, in chapter 4, look at access control, where we first present some basic theory of access control, before we then review different types of access control. In this chapter we will also give information about hash algorithms; theory, types, comparisons, and look at the iButton technology for use in our scenario. In this part we will give an introduction to iButtons, present information about different types, alternatives, and security issues relating to iButtons.

Chapter 5 will present the reader with information about dynamic routing. First we will discuss a selection of dynamic routing protocols for mobile ad-hoc networks, and then continue with a performance test of the protocol we selected for FieldCare. In the end we discuss security related to ad-hoc networks and protocols.

The implementation chapter will then discuss what we have implemented into the new FieldCare prototype. Here you will be presented with Unified Modeling Language (UML) sequence, activity, and class diagrams.

At the end of the report you will find a list of acronyms and a reference list.

In appendix A-C we have put our installation and configuration guides, results from our OLSR test, and miscellaneous scripts we have used.

2 FIELDCARE

This chapter provides a fundamental introduction to the FieldCare system and also SINTEF's visions for the future development of this project. Most of the FieldCare relevant information has been extracted from a SINTEF presentation [GSW02]. In addition we will present our suggestion for a new prototype with extended functionality.

2.1 BACKGROUND

SINTEF is an independent, non-profit, research institute with its base in both Oslo and Trondheim, and is organized in several different departments. This thesis was given by the department of Telecom and Informatics.

2.1.1 SINTEF TELECOM AND INFORMATICS

SINTEF receives only a very small portion of its income in the form of government grants, and depends on contract research with Norwegian and international industry for the bulk of its income. A part of this income is used to fund internal research in areas that have promising commercial prospects. The FieldCare project, which was started at SINTEF Telecom and Informatics in the summer of 2001, is an example of such a project. So far, all the work on the project has been paid from SINTEF's own funds, and has consisted of the following main activities:

- Development of the overall concept for supporting emergency care.
- Development of a prototype implementation, which ended up in a robust working solution (simply referred to as the FieldCare prototype).
- Contact with potential users, to describe/discuss solutions and requirements. This work has been very informal and based entirely on the goodwill of the organizations involved.

2.1.2 THE EMERGENCY SCENARIO

There are several coordinator-roles at the scene of a major accident. The leader of the medical team is responsible for the quality, planning and execution of the medical work. The top leader of the rescue operation is usually from the police. In big trauma situations, fire brigades and other resources might also be involved. All these coordinators need to cooperate and need to control and optimize the use of their resources. They need to take qualified decisions based on up-to-date information, but today this information is not easily available.

Broadcast radio networks are the main form of communication and information distribution used in emergency scenarios today. For the rescue personnel involved, it may be difficult to listen in on both the police and the medical radio network at the same time, as these two networks use different terminals.

A paper form with a unique id-number is attached to every patient on accident site (displayed in Figure 2.1). On this form, the patient priority is noted together with other important information, such as; injury, preliminary diagnosis and treatment. Today the paper form is mostly used to identify the patients and to some extent keep track of where they are. The id-number is printed on several tear-off tags which is supposed to be torn off and left at the places a patient goes through.

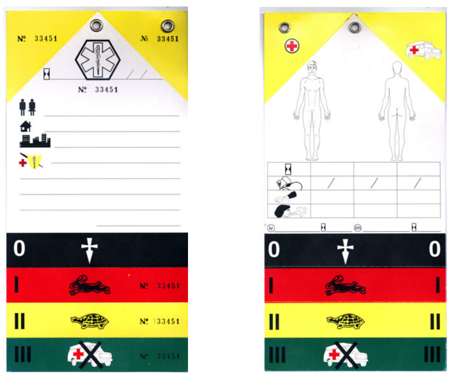


Figure 2.1 – An example of a paper form
Information may be added to both the front and back of the paper form, and the priority (0-3) is categorized in different colors.

Ambulance personnel have informed SINTEF Telecom and Informatics that the information noted on the forms by medical personnel is used to very little extent. There are no procedures on how to store the forms on arrival at the hospital. This might be due to an inherent culture amongst doctors; they always want to do a firsthand examination of a patient for themselves, irrespective of the previous examinations done by other medics. An additional problem with the paper form is that the information is attached to the patient, and the hospital will first get this information when the patient has arrived.

All medical and ambulance personnel are bound by law to document everything about their mission, and specifically the time at which certain things are done, and also who did it, is very important; e.g. the time arrived at the patient, amount of time worked with the patient, what was done to the patient, etc.

2.2 THE FIELDCARE PROTOTYPE

To ease the tasks performed by the rescue personnel at the accident site, as described above, SINTEF Telecom and Informatics developed the FieldCare prototype. They envision that medical personnel will carry mobile computers or tablet PC's with the FieldCare application, to assist them in their work. Such mobile devices should become part of the medics' everyday toolkit, with functions and procedures that support them throughout their work. The users must be well familiarized with the system before they take it into an emergency scenario. The interface to the user has to be easy to use, and in addition SINTEF believe that the emergency scenario is well suited for speech recognition, as the vocabulary is well defined.

The computers connect together in a wireless ad-hoc network at the scene of the accident using short-range radio networks, and are connected to transport units and hospitals by a long-range radio network. The core functionality of FieldCare is to enable data-capture at the point of origin and to replicate this up-to-date information to all people concerned at all sites, as shown in Figure 2.3.

The FieldCare prototype provides the means for instant messaging between the units on the accident site and has the framework set up for integration of FieldCare with other transport and hospital information systems.

A new concept was introduced in the FieldCare prototype, called *Medical Information Tag* (MiTag). iButtons was used to implement the MiTag. The MiTag has three functions:

- It identifies patients, as the medic will assign a MiTag to every new patient found. All MiTags are pre-initialized with a unique identification number.
- It is used as backup for the patient data in the system. If the network fails, a medic can still gain access to the entered patient data by reading the MiTag with the mobile computer.
- It can be pre-worn and carry static medical data like name, blood type and allergies. This can aid the emergency treatment. This is most probable in a military scenario, where all soldiers carry a tag with visual id.

The FieldCare network should be designed to be robust, so that error situations are handled in such a way that the user does not notice it. If a device loses network connection for some reason, the medic should be able to enter data as normal, and the system should automatically replicate the data when the connection is re-established.

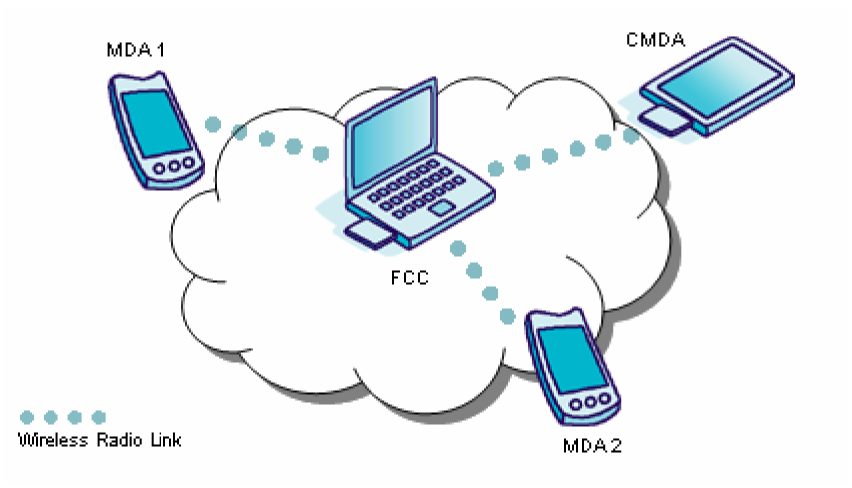


Figure 2.2 – The static network topology of the FieldCare prototype
Currently, the FieldCare prototype uses a static network topology configuration as shown in this figure. However, it is important to realize that this configuration does not constitute a single point of failure, as all data are replicated to every device (definitions are found in section 2.4).

Using the MiTag with FieldCare the information could arrive at the hospital almost at the same time it is generated, giving hospital personnel more time to prepare for the arrival of the injured. In addition, the MiTag will help the medical personnel by providing a running update and tracking of patients.

FieldCare is designed to be terminal independent, so that replacement of terminals is easy. In the FieldCare prototype, the same application runs on all devices, the only difference being the Graphical User Interface (GUI) module that tunes the GUI to the individual device characteristics.

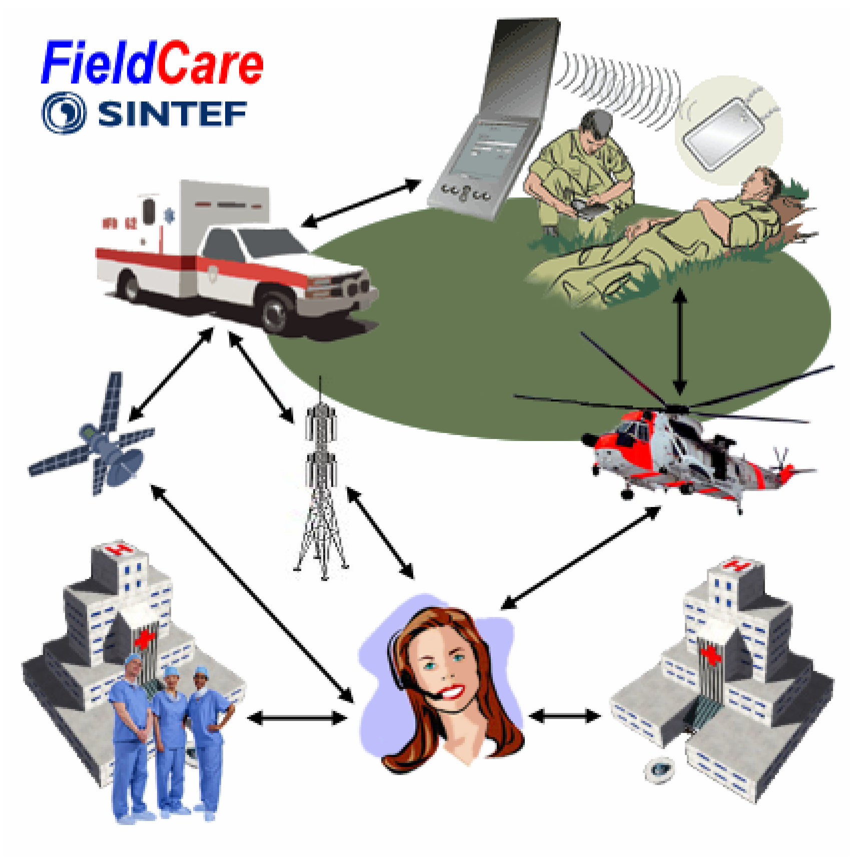


Figure 2.3 – FieldCare vision overview
FieldCare is a system using mobile devices, medium/long range radio links and smart software to gather information and share it between all people involved in dealing with a medical emergency [GSW02].

2.3 OUR NEW SYSTEM PROTOTYPE

Our suggestion to a new prototype is presented in this section, and it is based on the FieldCare prototype developed by SINTEF Telecom and Informatics.

The specific functional elements added to the existing FieldCare prototype are those specified in section 1.1; access control mechanism and a dynamic routing scheme, and also the possibility to securely download patient journals.

The figure below presents a schematic overview of our new system prototype, with elements from both our and Group 2's implementation. Please refer to section 2.4 for an explanation of each individual device.

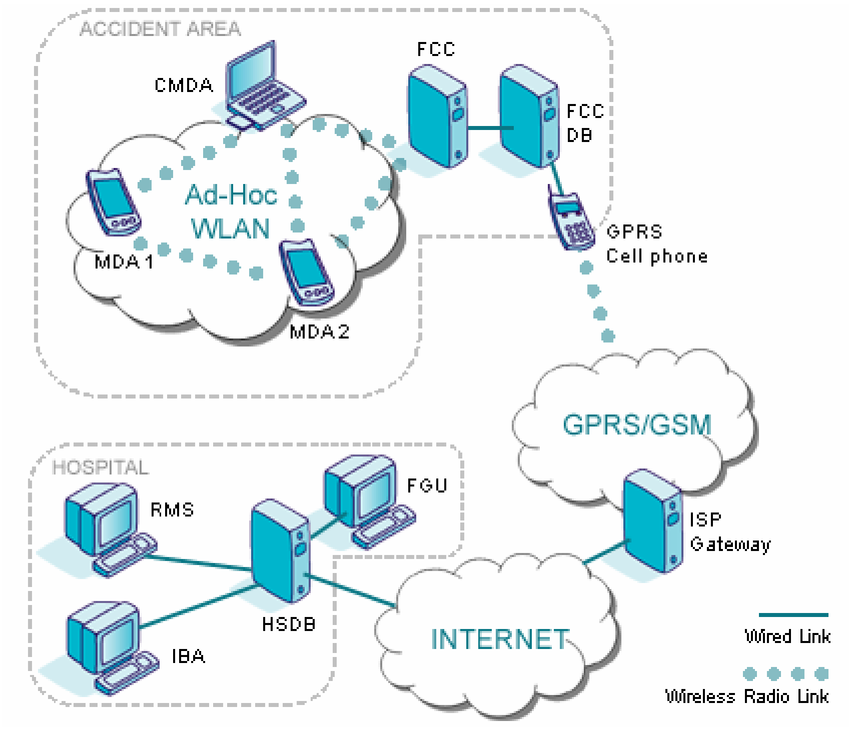


Figure 2.4 – The new system prototype
This figure presents a complete overview of the new system prototype. There are several new components in this network, and they are all explained in detail in section 2.4.

The following list is a brief summary of our implementations:

- A new access control mechanism using Personal Identification Tags (PiTag).

  A set of valid PIN codes were hard-coded into the FieldCare prototype. This made the addition of new users a non-trivial problem as the entire application would need to be recompiled each time new users were added. We have now designed and implemented a new access control mechanism, using PiTags containing user specific information (e.g. name, unit, language, etc.) and a hashed value of the user's PIN code.

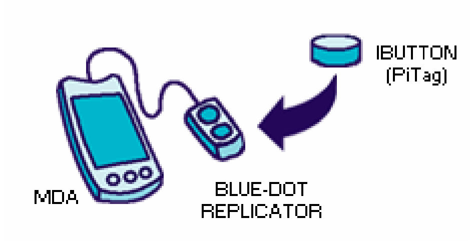


Figure 2.5 – iButton access control
Each device is equipped with an iButton reader (the Blue-Dot Receptor) and the personnel using the devices are given PiTags.

- A dynamic routing scheme for the ad-hoc network.

  The FieldCare prototype used a static routing scheme, where all participating nodes had to be predefined in a certain topology. Figure 2.2 shows the star-topology that was used. A dynamic routing scheme has now been implemented on the devices in the WLAN, creating a scaleable and highly adaptive network topology.

- The use of Competence Roles for access to electronic patient journals on the hospital.

  Authenticated users will now receive a Competence Role (defined in 2.4) that governs their information access level. This Competence Role is used when patient journals are downloaded from the database at the hospital, where this database will only extract the amount and type of information granted by the access level associated with the Competence Role.

Our implementation is more thoroughly described with the use of UML sequence, activity and class diagrams in chapter 6.

2.4 ACRONYMS AND CONCEPTS

Our new system prototype is packed with acronyms; therefore we present here a definition of the devices and concepts used, both those already defined in the FieldCare prototype and also those specific to our new system prototype.

MiTag – Medical Information Tag	The MiTag device is an electronic *tag*, attached to the patient at the scene of accident, and is used to store the most vital medical information about the patient. The MiTag is a highly dependable information source even if other systems should fail [SIN02]. In a military scenario, the MiTag can also carry preloaded medical information. Currently the iButton technology is used to represent this device.
PiTag – Personal Information Tag	The definition of the PiTag is almost identical to the MiTag; however the PiTag is used by the personnel on the accident site. The PiTag contains information about the user (i.e. name, title, unit, etc.) and it also contains a hashed value of the user's PIN code. This information is used for the authentication of the user during login, and during external requests (i.e. downloading of patient journals).

MDA – Medical Digital Assistant	The MDA device is a small handheld computer, similar in functionality of a Personal Digital Assistant (PDA), but the MDA is in addition equipped with hardware and software specialized for the tasks carried out by medical personnel in emergency situations [SIN02]. The MDA device can access both MiTags and PiTags.
CMDA – Coordinators Medical Digital Assistant	The CMDA is a portable computer with software to assist medical team leaders in carrying out the task they are responsible for. The CMDA is somewhat larger than the MDA, in order to provide a bigger screen area to facilitate a presentation of more overview information [SIN02].
FCC – FieldCare Connector	The FCC device is a computer, located in a transport unit (e.g. the ambulance), with display and radio communication equipment (short and long range), that provides the CMDAs and MDAs with a connection to other transport units and hospitals. All the data gathered by and stored on MDAs and CMDAs is replicated to the FCC [SIN02].
HSDB – Hospital Server Database	This database is located at the hospital and contains all the data regarding an accident (specific information on what has happened and updated information about the patients). The HSDB does also include information about the users of the FieldCare system, their Competence Roles and also patient journals [HMS03].
FCCU – FieldCare Connector Update	The FCCU is an application responsible for replicating all the data collected from the accident site to the HSDB. It is also responsible for the handling of various requests made by the devices in the WLAN (e.g. request for Competence Roles and patient journals) [HMS03].
FCCDB – FieldCare Connector Database	The FCCDB is a database that stores all the patient data entered by the medical personnel, and caches this data before it is replicated to HSDB by the FCCU application. This caching solution renders the system fault tolerant if the communication link to the hospital should be lost [HMS03]. Currently it is the FCCDB that has the actual interface to the external network, while the FCC has the interface to the internal network.

IBA – iButton Admin	The IBA system manages the handling and distribution of iButtons which are used as PiTags. The IBA should be located on the hospital's perimeter, for easy and secure access to the HSDB. The IBA system registers authenticated user data on a PiTag, and after the user (e.g. a doctor, paramedic etc.) has entered his/her PIN, the hashed value of this PIN is also stored on the PiTag.
FGU – Fast Graphical Update	This device is a graphical interface to the HSDB and it is consecutively updated during an accident. The FGU displays the current status of the accident and the patients involved, and it may be used by the Emergency Medical Communication Centre (EMCC) for resource allocation and planning [HMS03]. The EMCC (Nor: AMK) is not part of FieldCare.
RMS – Role Management System	The RMS is an interface to the HSDB device for the assignment and management of access levels in relation to the Competence Roles [HMS03].
Competence Role	A Competence Role specifies the access level granted to a user that has successfully logged on to a device in the ad-hoc WLAN. The Competence Role is issued by the HSDB and is also used by the HSDB to differentiate (i.e. filter) the information (e.g. a patient journal) sent back to the device on the accident site. A level 1 user will have access to the most basic data, e.g. the blood type, while a level 4 user will have access to highly sensitive data such as mental disorders and sexual diseases. The Competence Role is related to a user's level of competence; hence a doctor will have a Competence Role with a higher access level than the Competence Role to e.g. a paramedic.
Emergency Login	An Emergency Login is a procedure used to allow users immediate access to the FieldCare application by circumventing the login procedure. In some situations this concept is also referred to as Blue-Light access. This type of access is necessary to allow any persons to provide aid in a emergency situation.

Table 2.1 – List of acronyms and concepts
Acronyms and concepts used in the FieldCare prototype, together with those used in our new system prototype, are defined in this list.

2.5 FOCUS OF WORK

As was stated in section 1.2, the description of this thesis was originally much less extensive, however it was extended and divided into two parts when it was realized that a total of five persons were interested in this project. We got the responsibility of carrying out the first part, which was described in detail in 1.1. However, we and the other group decided to work in close cooperation, and together develop a new system prototype. This would yield a more complete and coherent end result. As displayed graphically in Figure 2.6, we focus on a different part of the system than Group 2. The FCC device is where our implementation merges with the work of the other group. There exists also collaboration at the hospital site, where we are in charge of creating the iButton Admin (IBA) system.

It is important to be aware of this work division, as we will often talk about the system as a whole, but will only be responsible for a subset of the system.

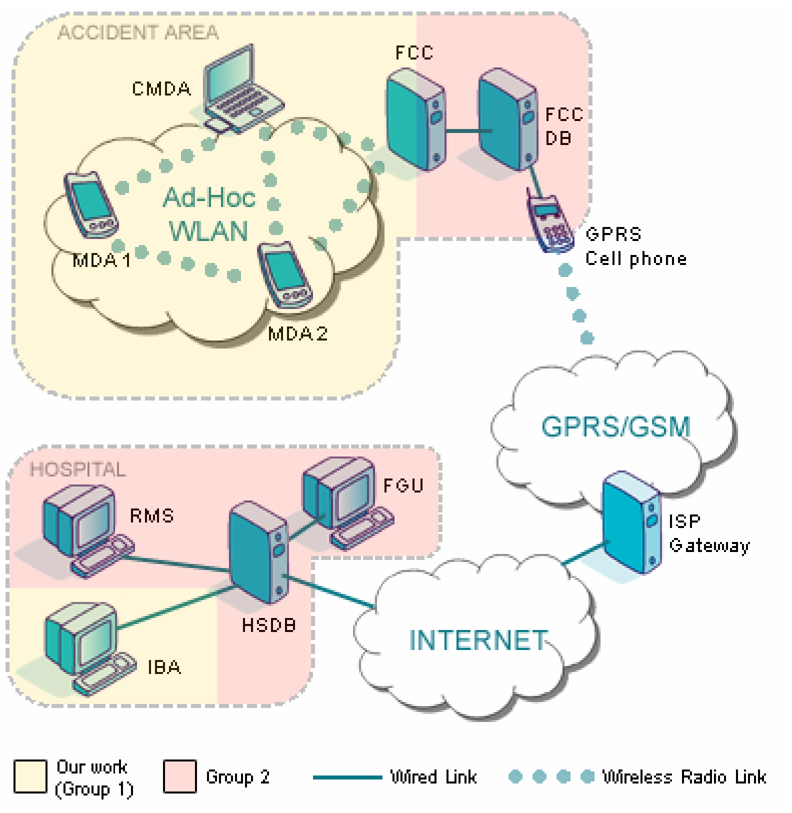


Figure 2.6 – Focus of work
This is a graphical representation of the focus of each group's work, superimposed on our final network layout.

3 SECURITY ANALYSIS

In FieldCare, an ad-hoc WLAN is deployed on the site of the accident and sensitive data flows in this network (e.g. personal data), therefore it is very important to be aware of the system's vulnerabilities and how to defend against possible attacks. A threat scenario is identified in section 3.1, where each identified threat is further analyzed in section 3.2 by using a risk assessment matrix. Since FieldCare involves personal data, we have to take into consideration the regulations and laws concerning the processing of such data. This is presented in section 3.3, where we have also contacted the proper authorities to get a qualified assessment of our interpretations. And finally in section 3.4 we present the security requirement specification for FieldCare.

3.1 IDENTIFYING THE THREATS

A given system is subject to many kinds of threats. A threat has the potential to cause an unwanted incident that may result in harm to a system and its assets. This harm can occur from a direct or indirect attack on the information being handled by an Information and Communication Technology (ICT) system or service, e.g. its unauthorized destruction, disclosure, modification, corruption, and unavailability or loss. A threat needs to exploit an existing vulnerability of the system in order to successfully cause harm to the system itself or its assets (e.g. sensitive data, applications, users etc.). Threats may be of natural or human origin and can be accidental or deliberate. To help identify and categorize the threats, we have used the ISO/IEC TR 13335 standard "*Information Technology – Security Techniques*" prepared by the Joint Technical Committee 1, subcommittee 27 of the ISO/IEC [Stu01].

Human		**Environmental**
Deliberate	**Accidental**	
Data modification	EMP	Earthquake
Denial of Service	Errors and omissions	Flooding
Device cloning	File deletion	Fire
Device theft	Incorrect routing	Lightning
Eavesdropping	Jamming	
EMP	Physical accidents	
Impersonation	Power failure	
Incorrect routing		
Jamming		
Malicious code		
Social engineering		
Traffic analysis		
War driving		

Table 3.1 – List of specific threats
The threats are categorized according to the specification given in [Stu01]. This list is open at the bottom to symbolize that we do not claim it to be complete and final.

Both the human based and the environmental based threats are identified, however we will only focus on the deliberate and accidental human based threats in our further analysis, since these threats are more relevant to this thesis. As for the environmental based threats, a quite substantial amount of statistical data is available regarding the frequency of occurrence and the level of damage [Stu01], and should be obtained and used by the entity developing the system for the final threat assessment process.

3.1.1 DATA MODIFICATION

This threat represents a deliberate and unauthorized insertion, alteration or deletion of data in a given system. Example of a specific attack is the *man in the middle attack* where the attacker has to position himself so that all traffic coming and going to the victim goes through him [Col01]. Then the attacker may perform any of the specified actions; insertion, alteration or deletion. To guard against this type of attack, the use of end-to-end encryption and strong authentication mechanisms with digital signatures are recommended.

3.1.2 DENIAL OF SERVICE

A Denial of Service (DoS) attack is an incident in which a user or a system is deprived of the services of a resource they would normally expect to have [SS03]. A more detailed definition and explanation of this threat is given in subsection 5.3.1.3, when we are specifically reviewing security threats in wireless networks. However a DoS attack can be represented in many various forms, e.g. a deliberate loss of power, as stated in 3.1.14 below, is an example of a DoS attack. As computer based DoS attacks usually exploit holes in software, networking practices, and operating systems, the best defense against this type of attack is by regularly patching, updating and upgrading the software used. However, in general, it is very difficult to create good protective measures against denial of service attacks.

3.1.3 DEVICE CLONING

Device cloning is the act of presenting false or duplicated credentials to gain access to a system resource; this threat is also called *masquerading*. Using strong authentication mechanisms with digital certificates and signatures will protect against most utilizations of this threat. Physical security mechanisms can also be applied to hinder device cloning. E.g. an iButton (presented in chapter 4.9) uses a shielding layer within the device to protect its unique identification number.

3.1.4 DEVICE THEFT

Device theft is the physical theft of any given device by an attacker. This is not a concept new or unique to ICT systems, but unfortunately, designing devices or systems resistant to theft is very difficult in general [SE03]. However all devices should be stored in secure locations when not in use. The probability of device theft is greater in mobile systems (e.g. wireless devices), as physical security is here more difficult to enforce.

3.1.5 EAVESDROPPING

Eavesdropping is an unauthorized interception and gathering of information by an attacker. This threat can easily be hindered by employing encryption mechanisms to the vulnerable system.

3.1.6 EMP

An electromagnetic pulse (EMP) is a very short but intense pulse, which is strongest at its source and weakening with distance. EMP is effectively an electromagnetic shock wave. It creates a powerful electromagnetic field that produce short lived transient voltages of thousands of Volts on any exposed electrical conductors, such as wires or conductive tracks on printed circuit boards. The effects of an EMP apply to all electrical and electronic equipment, and the damage caused by EMP can look almost like the results of a lightning strike.

Commercial computer equipment is particularly vulnerable to EMP as key components include high density Metal Oxide Semiconductor (MOS) devices, which are very sensitive to exposure to high voltage transients [Kop93]. An EMP attack can be represented either directly, in the form of an e-bomb, or indirectly as e.g. nuclear bombs produces an EMP as a byproduct when detonated.

Protection against the effects of EMP are extremely difficult, but [USA02] suggests that a proper grounding of all vulnerable components in a system, and the use of EMP barriers (a form of shield) should provide a low level defense against EMP attacks.

3.1.7 ERRORS AND OMISSIONS

Most of the software developed today and in the past are vulnerable to various design, implementation, programming, or usage faults, e.g. buffer overflows and improper use of memory management. These errors can lead to an unexpected and unwanted software behavior (bugs). Most software bugs can be remedied by using a good design and programming practice together with formal methods for validation and verification of the software.

3.1.8 FILE DELETION

Inexperienced users of a given system may accidentally delete data vital for a correct software execution. Deliberate file deletion is rather defined as part of the threat *data modification* in 3.1.1 above. Accidental file deletion may be prevented with proper training of the personnel using the system/device, but also by using authorization mechanisms and integrity tools.

3.1.9 IMPERSONATION

Impersonation is an act whereby an entity assumes the identity and privileges of another entity. As stated later in 5.3.1.5, impersonation is a form of spoofing attack. The use of strong authentication mechanisms is an effective defense against this form of attack.

3.1.10 INCORRECT ROUTING

By injecting erroneous routing information, replaying old routing information, or distorting routing information, an attacker could successfully partition a network or introduce excessive traffic load into the network causing retransmission and inefficient routing. Compromised nodes in a network may advertise incorrect routing information to the other nodes, and possibly creating a unauthorized change to the network topology. Specific examples of this threat are among others; the *black hole attack* and the *wormhole attack*. These two attacks are presented in more detail in subsections 5.3.1.6 and 5.3.1.7. Proper authentication of each node and the use of temporal tags (also called packet leaches and are later explained in 5.3.1.7), are ways to prevent incorrect routing information.

3.1.11 JAMMING

[T1A101] has a very good definition of jamming, which we have used again in 5.3.1.4. Jamming is the deliberate radiation, re-radiation or reflection of electromagnetic energy for the purpose of disrupting enemy use of electronic devices, equipment or systems. In some contexts, jamming is considered a special form of a DoS attack, and as with the DoS attack, jamming is also very difficult to defend against. However, it fairly easy to trace the jamming signal, hence locate the source of the attack.

3.1.12 MALICIOUS CODE

Malicious code, according to [T1A101], is software or firmware capable of performing an unauthorized function on an information system. Specific examples of malicious code are a virus, a worm or trojans. A virus is a program which, when executed, can add itself to other programs without permission, in such a way that the infected program can add itself to even more programs. A worm, on the other hand, is a program which copies itself into nodes in a network without permission. And a trojan is a program which masquerades as a legitimate program, but does something other than what was intended. A general protection against these types of malicious code is by using some sort of security reference monitor with access control mechanisms or enforcing integrity checks of the system.

3.1.13 PHYSICAL ACCIDENTS

All systems, that are in part or completely represented by and dependent on physical objects, are subject to physical accidents. And similar to the device theft threat, as stated in 3.1.4, when the level of mobility increases the probability of this threat also increases. This threat is usually mitigated simply by adding redundancy, e.g. in the form of extra backup devices, or by adding a protective shielding to the device making it more resilient.

3.1.14 POWER FAILURE

Systems that are dependent on electricity to operate properly are all vulnerable to faulty power supplies and power failures. This threat is often due to unmanageable elements, e.g. lightning strikes, but also due to technological problems, e.g. low battery capacity. Deliberate power failures can be categorized as a Denial of Service attack as presented in 3.1.2.

3.1.15 SOCIAL ENGINEERING

Basically, social engineering is the art and science of getting people to comply with the attacker's wishes [Har97]. Social engineering is based on the manipulation of the natural human tendency to trust, and is generally agreed upon as being the weakest link in any security chain [Gra01]. To protect a system from the use of social engineering it is important to have proper security policies defined and enforced. But it is also important to teach the targeted people in basic computer security and explain the problem regarding social engineering.

3.1.16 TRAFFIC ANALYSIS

The process of monitoring the nature and behavior of traffic, rather than its content, is known as traffic analysis [Sch95]. Traffic analysis usually works equally well on encrypted traffic as on unencrypted traffic. This is because common encryption methods, such as Secure Sockets Layer (SSL) and Data Encryption Standard (DES), do not try to obfuscate the amount of data being transmitted. Another definition of traffic analysis is presented in subsection 5.3.1.2, where we discuss various threats to wireless networks. Data padding, communication hiding schemes and selective rerouting have been suggested as a protection against the use of traffic analysis [NMS+03].

3.1.17 WAR DRIVING

War driving is the act of locating and possibly exploiting connections to wireless local area networks (WLAN) while driving around in a city or elsewhere [SS03]. Attackers can use a simple laptop computer with a wireless network interface card (NIC) set in a promiscuous mode and some kind of antenna mounted on the top of or positioned inside a vehicle. Software is readily available for free on the Internet, e.g. *NetStumbler* [Mil02] and *AirSnort* [SHB03]. The use of encryption and strong user authentication are good preventives against war driving.

3.2 HAZARD RISK ASSESSMENT MATRIX

In this section we analyze the deliberate and accidental human based threats presented in section 3.1. By using a hazard risk assessment matrix, we are able to categorize and define a risk level for each threat. The risk matrix we have used is based on the definitions in *Part 3* of the IEC 300-3-9 standard, "*Risk Analysis of Technological Systems*" [IEC95], and on the works of Communication-Electronics Command (CECOM) of the United States Army [USA99].

3.2.1 THE MATRIX

The risk matrix is based on two fundamental properties of any threat; the *probability* (the frequency of occurrence) and the *consequence* (the level of severity). The probability is categorized in five groups and CECOM have assigned these groups the codes A through E respectively; frequent, probable, occasional, remote and improbable. The consequence has four different levels of severity, where level 1 being the most critical. By combining the hazard probability with the hazard severity, we're able to assign each threat with a risk level (high, medium or low). [USA99] has also defined a hazard risk assessment code (RAC) for each element in the matrix.

HAZARD RISK ASSESSMENT MATRIX				
HAZARD PROBABILITY	HAZARD SEVERITY			
	I	II	III	IV
FREQUENCY OF OCCURRENCE	CATASTROPHIC	CRITICAL	MARGINAL	NEGLIGIBLE
(A) FREQUENT	**1A**	**2A**	**3A**	**4A**
(B) PROBABLE	**1B**	**2B**	**3B**	**4B**
(C) OCCASIONAL	**1C**	**2C**	**3C**	**4C**
(D) REMOTE	**1D**	**2D**	**3D**	**4D**
(E) IMPROBABLE	**1E**	**2E**	**3E**	**4E**
Hazard Risk Assessment Code (RAC)	Risk Level			
1A, 1B, 1C, 1D, 2A, 2B, 2C, 3A	High			
1E, 2D, 3B, 3C, 4A	Medium			
2E, 3D, 3E, 4B, 4C, 4D, 4E	Low			

Table 3.2 – Hazard risk assessment matrix
The matrix maps a threat's frequency of occurrence with it's severity level, and assigns a Risk Assessment Code to each combination.

If we then use the threats we identified in section 3.1, it is now possible to assign a risk level to each specific threat by using the above risk matrix.

Threat	**RAC**	**Risk Level**
Data modification	2D	🟧 Medium
Denial of Service	1C	🟥 High
Device cloning	3D	🟨 Low
Device theft	3C	🟧 Medium
Eavesdropping	3C	🟧 Medium
EMP	1E	🟧 Medium
Errors and omissions	3B	🟧 Medium
File deletion	3C	🟧 Medium
Impersonation	2E	🟨 Low
Incorrect routing	2C	🟥 High
Jamming	1E	🟧 Medium
Malicious code	2D	🟧 Medium
Physical accidents	3B	🟧 Medium
Power failure	3C	🟧 Medium
Social engineering	2D	🟧 Medium
Traffic analysis	4D	🟨 Low
War driving	3C	🟧 Medium

Table 3.3 – Risk level assignment
The human-based deliberate and accidental threats presented in section 3.1 are in this table assigned a risk level from the risk matrix in Table 3.2.

3.2.2 DISCUSSION

Defining a RAC for each threat is the essential element here, but unfortunately there are no formal methods for selecting this value. Therefore it must be assigned selectively for each scenario by those responsible for the risk analysis. Experience is a key issue in this process; hence it would be imprudent of us to claim that our evaluation of the threats is flawless, however it is done to the best of our knowledge.

Since the ad-hoc WLAN is probably deployed for only a limited amount of time (hours or a few days), some of the threats that are highly time dependent (e.g. traffic analysis), will have a low frequency of occurrence. This will also to some extent apply to other threats, as the time exposed (i.e. the time the system is vulnerable to attacks), is kept at a minimum. Therefore many of the threats in Table 3.3 were assigned a low hazard probability (D-E). There are also other considerations to make, for instance in a civilian emergency scenario, it is unlikely that attacks like EMP and jamming will be used, as these apply more to modern warfare. However, these two attacks have such a high level of severity, that the resulting risk level is still medium.

Due to the vast amount of available tools to execute a DoS attack, and the simplicity of using these tools, the DoS threat was assigned a probability C (i.e. occasional), despite of the limited exposure time as described above. As all the devices in the ad-hoc network have routing capabilities, a DoS attack may severely disrupt the processing capacity of the devices, thus depriving the medics of the ability to enter patient data and communicating with other devices. A successful DoS attack will therefore have catastrophic consequences for the operation of the WLAN. In addition, the coordinators on the accident site will not receive any information, and hence they are unable to properly plan and execute the medical work. Because of the high level of severity, the threat of a DoS attack was assigned a high risk level.

Incorrect routing was also assigned a high risk level, due to similar reasons as with the DoS threat. However in this case, the medics are still able to enter patient data locally on their device, but due to the incorrect routing, all the communication between devices are greatly disrupted, and the coordinators may not receive the necessary data to properly perform their tasks. Since patient data can be stored on both MiTags and locally on each device, and is ready to be distributed to all devices if the threat is eliminated, we categorized this threat as critical, and not catastrophic.

3.3 SECURITY AND PRIVACY REGULATIONS

In Norway, the Data Inspectorate (Nor: Datatilsynet) is the regulatory agency responsible for enforcing the laws of data privacy and protection. The Data Inspectorate, which is an independent administrative body under the Norwegian Ministry of Labour and Government Administration, was created in 1980 to ensure the enforcement of the *Data Register Act* of 1978, now made obsolete by the commencement of the *Personal Data Act* of 2000.

The purpose of this Act is to protect persons from violation of the rights to privacy through the processing of personal data. The Act shall help to ensure that personal data are processed in accordance with the fundamental respect for the right to privacy, including the need to protect personal integrity and private life, and to ensure that personal data are of adequate quality [TDI03].

Information security in the context of personal and medical data as used in FieldCare is governed by the common regulation of the two following laws; *The Personal Data Act* (Nor: Personopplysningsloven) and *The Health Registry Act* (Nor: Helseregisterloven).

3.3.1 THE PERSONAL DATA ACT AND THE SV-100:2000 REGULATION

As of January 1st 2001, the Personal Data Act of 2000 [PDA00] replaces the Data Registry Act of 1978. In addition to this law, the Data Inspectorate has compiled an information security regulation (SV-100:2000) regarding the use and processing of personal data [SV100]. This regulation is only applicable to situations where personal data are in part or completely processed electronically (e.g. FieldCare), otherwise the general rules specified in § 13 of the Personal Data Act should be used. The system in question has to be in compliance with both the Personal Data Act and the SV-100:2000 regulation before the use and processing of personal data may commence.

3.3.2 THE HEALTH REGISTRY ACT

The purpose of the Health Registry Act [HRA01] is to provide information and knowledge to the health care service industry and the health care administration, without violating personal privacy, such that care can be administered in a safe and efficient manner. The law shall ensure that medical information is processed in accordance with fundamental considerations to privacy protection, integrity and information quality.

3.3.3 REGULATIONS APPLICABLE TO FIELDCARE

In the introductory chapter, section 1.2, we specified that this project is actually divided between two separate groups; us (Group 1) and Group 2. Even though both groups work within the same FieldCare scenario, each group has focused on different areas of the system. In this section we have concentrated on the regulations securing the communication in a medical scenario, i.e. we have examined the Personal Data Act instead of the Health Registry Act, which is more applicable to Group 2.

We contacted, Mr. Bjørn Nilsen, a senior engineer at the Data Inspectorate, in order to get a qualified review of the interpretations we have done on the most relevant paragraphs in the regulation of the Personal Data Act [SV100]. Our interpretations and questions, and the replies from Mr. Nilsen are presented in the subsections below[1].

3.3.3.1 THE CONSIDERATION OF HUMAN LIFE

According to § 2-1 in the SV-100:2000 regulation;

"*The rules in this chapter apply to the handling of personal data when using, in part or completely, electronic facilities, where it is necessary to ensure the confidentiality, integrity and availability of the data in order to prevent any risk of human loss,*

[1] We disclaim all liability for the translation of the regulations in this chapter to English. The original document source [SV100] should always be consulted.

financial loss or loss of human reputation and integrity. Wherever such risk exists, the planned and formal initiatives being executed and upheld by the regulation shall be in relation to the probability of security breaches and consequences of such."

The Data Inspectorate's official comment to § 2-1 as stated in the SV-100:2000 regulation:

"*[...] The restraints in § 2-1 are important to have, in order to avoid establishing too extensive security measures.*"

Our question to Mr. Nilsen regarding § 2-1 was:

"*Is it correct to say that the consideration of preserving human life overshadow the consideration of security and personal protection in any medical context?*"

In the reply, he stated that there is, in principle, no element of conflict in this context. Information security shall comprise confidentiality and integrity as well as the availability aspect. That means that information must be secured with respect to confidentiality, but in certain special situations (often called blue-light situation), it may be advisable or even necessary to let the confidentiality requirement yield for the consideration of availability.

3.3.3.2 COMMUNICATION IN EMERGENCY MEDICAL SCENARIOS

The National Centre on Emergency Health-Care Communication (KoKom) in Norway has released a procedural handbook for the communication in emergency medical scenarios. In chapter 8, section 2, of this handbook, the following is stated about communication procedures in the Norwegian medical radio network (Nor: Helseradionettet):

"*Any radio based network, including NMT 450/900 and to a certain degree also GSM, are vulnerable to eavesdropping. With the exception of emergency situations, the medical radio network is not suited for the distribution of identifiable patient data.*"

In addition to this, in same section, KoKom states the following:

"*In an emergency situation (red), all necessary information must be transmitted in plain speech. [...] In urgent (yellow) or normal (green) situations, the use of a voice scrambler device is recommended.*"

It would appear that this is not in accordance with the regulations from the Data Inspectorate. § 2-11 in the SV-100:2000 regulation states:

"*Personal data that are electronically transferred using a transfer medium outside of the physical control of the unit responsible of processing the data, must be encrypted or equivalently secured when confidentiality is required.*"

The Data Inspectorate's official comment to § 2-11 as stated in the SV-100:2000 regulation:

"*The responsible unit shall by encryption or equivalent measures ensure the confidentiality of personal data transferred in a public communication network. The rules for encryption applies also to transfers through private communication links which is outside of the area secured against unauthorized access by the responsible unit, cf. § 2-10 of the SV-100:2000 regulation.*"

Our question to Mr. Nilsen regarding the KoKom procedure and § 2-11 was:

"*How should we regard the procedures presented by KoKom in the FieldCare scenario? And is there any disagreement between the KoKom procedure in section 8.2 describing discretion and § 2-11 in the SV-100:2000 regulation?*"

In the reply from Mr. Nilsen he explains that information security in our scenario, in general, is governed by the common regulation of the Personal Data Act and the Health Registry Act, and no other place. There may exists routines and procedures created by various institutions and organizations (e.g. KoKom), but in relation to the authorities' supervisory competence for information security in this area, the specified regulation will always be used a basis. Regarding our second question on whether or not there exists a disagreement, Mr. Nilsen states that it will not be satisfactory to build an information system for emergency medical situations where confidentiality has not be given the appropriate consideration. The law requires that there exists planned and formal efforts to ensure a satisfactory information security. When designing a radio based system for the transfer of personal and medical data, one is required to consider the confidentiality requirement and create measures for a satisfactory security, and Mr. Nilsen is not aware of any better measure than encryption for providing such security.

The FieldCare application is also envisioned to be able to request patient journals to be transferred directly from a hospital and out to the scene of the accident. And as we interpreted the regulations for this, cf. § 2-11, the use of some form of end-to-end encryption is required.

Mr. Nilsen verifies that our interpretation is correct, and in addition to the end-to-end encryption, one must also clarify which persons/devices are allowed to request such information by using e.g. some authentication mechanism.

We contacted also Mr. Magne Lillebø, a telecom engineer and a senior consultant for KoKom, who can verify that all the procedures specified in the KoKom manual are only applicable to the current analogue medical radio network. If a new digital network, consisting of other technologies, was to be constructed, the procedures would have to be modified, states Mr. Lillebø.

3.4 FIELDCARE SECURITY REQUIREMENT SPECIFICATION

The following security requirement specification has been compiled by considering the threats identified in 3.1 together with the result from the risk analysis in 3.2 and the regulations specified by the Data Inspectorate in 3.3. The requirements apply in most part to the ad-hoc WLAN, as this is our main focus in this thesis, however some do also apply to the system as a whole.

Secure PiTag Management	The management and distribution of PiTags must be secured. This means when a PiTag is issued, the user requesting it must be authenticated (e.g. by having the user provide valid identification credentials). The issuing entity must also be located in a manageable, secure and closed environment (e.g. being a part of the hospital), and be manned with entrusted personnel.
Secure MiTag and PiTag Data	The MiTags and PiTags are both used to store sensitive information; therefore these devices must store this data in a secure manner. The MiTags may contain patient data, and if this data is able to identify the patient it must be kept confidential by using e.g. encryption. The PiTags contain information applied in the authentication process of the user during the access control on the computer units (CMDA, MDA and FCC), therefore the data vital for authentication must be kept confidential.
Secure Device Deposit	To prevent device theft and unauthorized device modifications, the MDAs, CMDAs and FCC devices must be kept in a secure physical location when not in use.
Durability	All mobile devices used outdoors, must be protected against general "wear and tear" and environmental conditions like rain, mud and dust, to prevent the loss of availability.
End-to-End Encryption	All communication sessions must use an end-to-end encryption on the transport layer or above. However, in certain special situations, often referred to as blue-light situations, it is allowed by law to downgrade the confidentiality requirement if it in any way restrains the availability requirement.
User Authentication	All users associated with the FieldCare system must be authenticated before access may be granted (with the exception of the Emergency Login functionality).
Restriction on Emergency Login	Since any user may login in emergency mode (i.e. no authentication), certain restrictions must be implemented. Emergency users should have no access to the external network and not be allowed to download patient journals directly from HSDB, nor through any other third party.

Secure Control Data	All control data (e.g. routing information) must be protected against any form of deliberate or accidental modification or deletion.
Maintaining Integrity	An integrity check should be performed on a regular basis on all devices in the FieldCare system. The integrity check should be able to verify the FieldCare application and the state of the system (i.e. check for code modification and the presence of malicious code).
Software Maintenance	All the software used on the devices in the FieldCare system should on a regular basis be updated and patched to remove the presence of possible exploits and other vulnerabilities.
Power Redundancy	All mobile devices (MDA, CMDA and FCC) should use redundancy in their power supply (e.g. extra batteries, Uninterruptible Power Supply (UPS) units, etc.) to maintain a higher level of availability.
Personnel Training	The users of the FieldCare system should be properly educated and trained specifically for this system, in order to minimize the possibility of security breaches due to incorrect use of the system.
Well Defined Procedures	The existence of well defined and documented procedures is paramount. A minimum requirement should be to have procedures for the handling of situations where devices have been stolen or misplaced, and also to have procedures for access removal (i.e. when a person is relocated, fired or deceased).

Table 3.4 – Security requirements
This list specifies, in a non-prioritized manner, various security requirements specifically for the FieldCare system. Both functional and non-functional requirements are specified.

3.5 SUMMARY

The identification of possible threats to the FieldCare system together with the analysis of these threats were done in accordance with recognized methods and standards for evaluation of security threats and risks, i.e. the ISO/IEC TR 13335 standard [Stu01] and the IEC 300-3-9 standard [IEC95]. However the assignment of hazard risk assessment codes (RAC) is still a selective process, which is dependent on the level of experience of the persons involved. In our discussion of the risk assignments, we categorized the threats of Denial of Service and Incorrect Routing as high level risks, mostly due to the severe consequences these hazards represents in the WLAN on the accident site.

When we reviewed the laws and regulations applicable for FieldCare, we quickly realized that it was in both our and SINTEF's best interest to get a professional qualitative assessment of the interpretations we did. We contacted the Data Inspectorate, and after a few queries we received some assistance from senior engineer Bjørn Nilsen. He presented a clear and unequivocal comment to our questions and interpretations regarding the FieldCare system, among other things he emphasized on the need for confidentiality on an end-to-end basis.

The KoKom organization was also contacted to clear up the confusion we had regarding the applicability of KoKom's procedures in the FieldCare scenario. Our presumption was verified by KoKom; their current procedures were only applicable to the medical radio network of today.

The security requirement specification we compiled was a composition (and abstraction) of the results from the threat identification process, the risk analysis and the review of the laws and regulations. We presented the FieldCare specific requirements in relation to the prototype developed by the cooperative effort of our group and Group 2, with emphasis on our part of the system.

4 ACCESS CONTROL

In FieldCare, access control is essential. We need to control access to the program itself on the different devices and what information a user is allowed to retrieve and view, based on his/her access level. The next sections present the theory of access control. When accessing a system, using a personal PIN code is often used and this PIN code is usually hashed to strengthen the security, e.g. hashing is used when logging on to your user account on your computer. Section 4.6 will present different types of hash algorithms we have looked at for the use in our access control implementation. We have looked at iButtons for the use as PiTags in our implementation, and the iButton technology is presented in section 4.9.

4.1 THE FIELDCARE LOGIN PROCEDURE

This section will present the login procedure used in the FieldCare prototype, and how to improve this procedure in our new system prototype.

The FieldCare prototype logged the user in without a satisfactory authentication mechanism. When using a PiTag, the user was immediately granted access to the application, otherwise the user could enter a PIN code that was compared to a hard coded value in the prototype.

To improve this login procedure, we will combine the use of PiTags with the use of a PIN code for authentication. IButtons is a product SINTEF Telecom and Informatics is using in their FieldCare prototype as MiTags, and we will use this technology to represent the PiTags in our implementation. The PiTags will be used to store user information and a hash-value of the user’s PIN code. Our implementation of this access control is explained in more detail in section 6.4.

As was stated in the security requirement specification (section 3.4), any user should be able to logon in emergency mode (i.e. no authentication), as it is necessary to allow any persons to assist in medical emergency situations. The FieldCare prototype has already implemented a solution for an Emergency Login; however certain restrictions should be added. We will use the concept of Competence Roles to give only authenticated users access to the external network and the possibility to download patient journals directly from the hospital. A Competence Role will not be given to users logged on in Emergency mode; hence a restriction will be enforced on these users. The Emergency Login will now be in compliance with the requirements given by the Data Inspectorate.

4.2 ACCESS CONTROL THEORY

Some form of access control exists in almost every network, even if it’s merely that of the native operating system. To protect different resources, networks might have several access control tools. The various types of access control tools enforce security policy and/or users’ privileges by protecting system resources. There are many types of access control tools and many non-standardized terms to describe them; hence the difficulty in determining which solution is appropriate for a given network.

Whatever the access control solution may be, there are only a few approaches to configuring access permissions. These approaches strive to achieve the common goals of a finer level of granularity (i.e. a more precisely defined set of permissions) and simplification of the access control management; usually sacrificing one goal to some extent in order to meet the other.

As the level of granularity gets finer, a user is *less* likely to be granted unnecessary or denied necessary access permissions. Likewise, the coarser the granularity is, the user is *more* likely to be granted unnecessary access permissions.

4.3 ACCESS CONTROL TYPES

This section will give a presentation of different access controls, and explain briefly how they work. [Cam01] was used as a basis for the definition of each access control presented below.

4.3.1 USER BASED ACCESS CONTROL

Identity Based Access Control (IBAC) is another name for User Based Access Control (UBAC). UBAC requires a system administrator to define permissions for each user, based on the individual's needs. This access control has the potential to result in more finely grained permissions, although an effective UBAC system would be so labor intensive that it would be cost-prohibitive. It is impossible for security management to know precisely what access each and every user needs, and accordingly configure the permissions, and update them daily to avoid a build-up of outdated entries.

In actual practice, UBAC has largely been implemented by designating extremely coarse user groups, giving each user far more access permissions than they could possibly need.

4.3.2 ROLE BASED ACCESS CONTROL

Role Based Access Control (RBAC) is popular because it intends to advance permissions configurations towards the common goals. RBAC entails the mapping of different "roles" in an organizational hierarchy and defines a profile of access permissions to the network's resources for each role. Then each user is assigned one or more roles, providing him/her with access permissions defined by those roles. A user with super-user access may be assigned to every role, whereas someone on a need-to-know basis may be assigned only one or two roles.

RBAC has the potential of refining the granularity by assigning specific types of privileges to specific resources. Users in a certain role may be granted access to *read* but not to *write* certain files.

4.3.3 POLICY BASED ACCESS CONTROL

Policy Based Access Control (PBAC) is also known as Rule Set Based Access Control (RSBAC). An access control policy is a set of rules that determine users' access rights to resources within an enterprise network. One prevailing mechanism for enforcing enterprise policy is the use of access control lists (ACLs). ACLs are associated with every resource, and lists the users or groups and their access rights.

When using ACLs to enforce a policy, there is usually no distinction between the policy description and the enforcement mechanism, i.e. the policy is essentially defined by the set of ACLs associated with all the resources on the network. Having a policy being implicitly defined by a set of ACLs makes the management of the policy inefficient and error prone. In particular, every time an employee leaves a company, or simply changes a role within the company, an exhaustive search of all ACLs must be performed, so that the user privileges are modified accordingly. It follows that such policies are usually defined with a very low granularity, and hence tend to be overly permissive.

In contrast, PBAC makes a strict distinction between the formal statement of the policy and its enforcement. Making rules explicit, instead of concealing them in ACLs, makes the policy easier to manage and modify. Such a mechanism is usually based on a description language which should be expressive enough to easily formulate the policy rules. Coupled with the description language of the policy, there should be an enforcement mechanism capable of intercepting access attempts, evaluating them against the policy, and accordingly granting or denying the access request.

4.3.4 CONTENT DEPENDENT ACCESS CONTROL

Content Dependent Access Control (CDAC) is a method for controlling access of users to resources based on the content of the resource. CDAC is primarily used to protect databases containing potentially sensitive data.

CDBAC involves a lot of overhead, which is a result from the need to scan the resource when access is to be determined. High levels of granularity are only achievable with extremely labor-intensive permission configuration and a continuous management.

4.3.5 CONTEXT BASED ACCESS CONTROL

Context Based Access Control (CBAC) is most commonly used to protect the data traffic through firewalls. Since *context* and *content* sound similar, many people confuse them, but these are actually two completely different approaches to access control, which may be used simultaneously. CBAC means that the decision on whether a user can access a resource is not solely dependent on who the user is, what resource it is, or even the content of the resource (as in the case of Content Based Access Control), but also on the sequence of events that preceded the access attempt. Configuration of permissions for specific users is not required. Strict security policy rules are set and form the basis of decisions to permit or deny access.

4.3.6 VIEW BASED ACCESS CONTROL

View Based Access Control (VBAC) primarily protects database systems; thus for files and other applications, VBAC is not a functional solution. As opposed to other notions of access control, which usually relate to tangible objects, like files, directories, printers, etc., VBAC perceives the resource itself as a collection of sub-resources. The granularity can be very fine, but permission configuration is incredible labor-intensive. In particular with VBAC, defining the resources is extremely complex. Such configuration requires intimate knowledge of the data structures and the mutual relationships between users and data.

4.3.7 MANDATORY AND DISCRETIONARY ACCESS CONTROL

One of the attributes by which an access control policy is classified is whether it is a Mandatory Access Control (MAC) policy or a Discretionary Access Control (DAC) policy. As the name implies, a MAC policy is obligatory, i.e. it dictates whether an operation should be permitted or denied without allowing a user to override the policy. A DAC policy, on the other hand, leaves final decision in the hands of the end-user. An access control policy does not have to be strictly mandatory or strictly discretionary. Strict security standards such as the Trusted Computer System Evaluation Criteria (TESEC) employed in military environments, require MAC polities to be in effect, but MAC policies are usually not used in the business world. This is not because they are not useful, but rather because they are practically impossible to implement using the standard tools of the operating system.

The finest level of granularity achievable with MAC is dependent on the ability to precisely define least privilege permissions for all users. DAC policies can in theory be very finely grained, but in reality, DAC is extremely labor intensive, as each user must define permissions for all users to every resource owned.

4.4 HASH ALGORITHM INTRODUCTION

What can be done to secure user passwords? It is possible to store user login information in a password-protected database. But if the database is compromised all the user login information is compromised as well. Even if the security of the database is good, all the information is still accessible to persons with super-user access. This could to a certain extent violate user privacy. To secure the login information further, it could be scrambled by using a home-brewed algorithm (perhaps a few character permutations and substitutions), but this would just add some obscurity and not real security. Using some a standard cipher, such as DES or RC2, to encrypt the login information, would greatly strengthen the security. But these algorithms require an encryption key, which has to be kept secret. If this key is compromised, all login information is also compromised, and again the user privacy is not protected.

By using a one-way hash algorithm, login information is secured and user privacy is protected. A hash function H is said to be one-way if it is hard to invert, i.e. given a hash-value h, it is computationally infeasible to find some input x such that $H(x) = h$. Login information is hashed and the resulting hash-value is stored instead of the actual information. When a user wants to log in, the submitted password is hashed and compared to the hash-value previously stored. If they match, the user it authenticated.

The following sections will provide information about hash algorithms. The next section is an introduction to what a hash algorithm is and the requirements for such an algorithm. Section 4.6 explains briefly how the algorithms we have examined work. In section 4.7 we will present some differences and similarities between these algorithms, and discuss our selected hash algorithm used in the new access control mechanism.

4.5 HASH ALGORITHM THEORY

A hash function has many names, among others; message digest, fingerprint and compression function. A hash function *H* is a transformation that takes a variable-sized input *m* and returns a fixed-sized string, which is called the hash-value *h* (that is, *h* = *H(m)*). In general *H(m)* will be much smaller in length than *m*; e.g., *H(m)* might be 64 or 128 bits, whereas *m* might be a megabyte or more. Hash functions with just this property have a limited usability, but when employed in cryptography the hash functions are usually chosen to have some additional properties. The basic requirements for a cryptographic hash function are as follows [RSA00].

- The input can be of any length.
- The output has a fixed length.
- *H(x)* is relatively easy to compute for any given *x*.
- *H(x)* is one-way.
- *H(x)* is collision-free.

It is theoretically possible that two distinct messages could be compressed into the same message digest, resulting in a collision. The security of hash functions thus requires collision avoidance. Collisions cannot be avoided entirely, since in general the number of possible messages will exceed the number of possible outputs of the hash function. However, the probability of collisions must be minimized. If, given a message *x*, it is computationally infeasible to find a message *y* not equal to *x* such that *H(x) = H(y)*, then *H* is said to be a weak collision-free hash function. A strongly collision-free hash function *H* is one for which it is computationally infeasible to find any two messages *x* and *y* such that *H(x) = H(y)*.

The last requirement guarantees that an alternative message hashing to the same value as a given message cannot be found. This prevents forgery and also permits *H* to function as a cryptographic checksum for integrity.

The hash value concisely represents the longer message or document from which it was computed; one can think of a message digest as a "digital fingerprint" of the larger document. Perhaps the main role of a cryptographic hash function is in the provision of message integrity checks and digital signatures. Since hash functions are generally faster than encryption algorithms, it is typical to compute the digital signature or integrity check of some document by applying cryptographic processing to the document's hash-value, which is quite small compared to the document itself. Additionally, a digest can be made public without revealing the contents of the document from which it is derived [RSA00].

4.6 DIFFERENT TYPES OF HASH ALGORITHMS

This section gives you a brief overview of the hash algorithms we have looked at and we will explain briefly how each algorithm work, without going into any details.

4.6.1 MD5

The Message Digest algorithm version 5 (MD5) [Riv92], was developed by Ronald L. Rivest at Massachusetts Institute of Technology (MIT). MD5 has been one of the most widely used secure hash algorithms, but due to the threat of both brute-force and cryptanalysis some concerns have arisen.

The algorithm takes as input a message of arbitrary length and produces as output a 128-bit message digest. The input is processed in 512-bit blocks.

MD5 works in 5 steps:

1. *Append padding bits*
   The message is padded so that its length in bits is congruent to 448 modulo 512 (length ≡ 448 mod 512). Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 512.
2. *Append length*
   A 64-bit representation of the length in bits of the original message (before padding) is appended to the result of step 1 (least significant byte first). If the original length is greater than 2^{64}, then only the low-order 64-bits of the length are used. Thus, the field contains the length of the original message, modulo 2^{64}.
3. *Initialize MD buffer*
   A 128-bits buffer is used to hold intermediate and final results of the hash function.
4. *Process message in 512-bit (16-work) blocks*
   The heart of the algorithm is a compression function that consists of four "rounds" of processing.
5. *Output*
   The output after the processing is a 128-bit message digest.

MD5 is more complex and hence a bit slower to execute than MD4. Rivest felt that the added complexity was justified by the increased level of security provided.

4.6.2 SHA-1

The Secure Hash Algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST) and published as a Federal Information Processing Standard (FIPS PUB 180) in 1993. A revised version was issued as FIPS PUB 180-1 in 1995 and is generally referred to as SHA-1. SHA is based on the MD4 algorithm, and its design closely models MD4 [NIST93, NIST95].

SHA-1 works in 5 steps:

1. *Append padding bits*
   The message is padded so that its length is congruent to 448 modulo 512. Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 512. The padding consists of a single 1-bit followed by the necessary number of 0-bits.
2. *Append length*
   A block of 64 bits is appended to the message. This block is treated as an unsigned 64-bit integer (most significant byte first) and contains the length of the original message (before padding).
3. *Initialize MD buffer*
   A 160-bits buffer is used to hold intermediate and final results of the hash function.
4. *Process message in 512-bit (16 word) blocks*
   The heart of the algorithm is a module that consists of four rounds of processing of 20 steps each.
5. *Output*
   The output after the processing is a 160-bit message digest.

4.6.3 RIPEMD-160

The RIPEMD-160 message digest algorithm was developed under the European RACE Integrity Primitives Evaluation (RIPE) project, by a group of researchers that launched partially successful attacks on MD4 and MD5. The group originally developed a 128-bits version of RIPEMD. After the end of the RIPE project, Hans Dobbertin (who was not a part of the RIPE project) discovered attacks on two rounds of RIPEMD, and later on MD4 and MD5. Because of these attacks, some members of the RIPE consortium decided to upgrade RIPEMD.

RIPEMD-160 was intended to be used as a secure replacement for the 128-bit hash functions MD4, MD5, and RIPEMD. This algorithm is a strengthened version of RIPEMD with a 160-bit hash result, and is expected to be secure for the next ten years or more. The design philosophy was to build as much as possible on experience gained by evaluating MD4, MD5 and RIPEMD. Like its predecessors, RIPEMD-160 is tuned for 32-bit processors [DBP99].

RIPEMD-160 work in 5 steps:

1. *Append padding bits*
   The message is padded so that its length is congruent to 448 modulo 512. Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 512. The padding consists of a single 1-bit followed by the necessary number of 0-bits.
2. *Append length*
   A block of 64 bits is appended to the message. This block is treated as an unsigned 64-bit integer (least significant byte first) and contains the length of the original message (before padding).

3. *Initialize MD buffer*
   A 160-bits buffer is used to hold intermediate and final results of the hash function.
4. *Process message in 512-bit (16 word) blocks*
   The heart of the algorithm is a module that consists of 10 rounds of processing of 16 steps each. The 10 rounds are arranged as two parallel lines of five rounds.
5. *Output*
   The output after the processing is a 160-bit message digest.

4.7 HASH ALGORITHMS – DIFFERENCES AND SIMILARITIES

All three hash algorithms MD5, SHA-1 and RIPEMD-160 are derived from MD4 and are quite similar to one another. Accordingly, their strengths and other characteristics should be similar. Table 4.2 highlights some of the differences and similarities. [Sta98] has been used as basis for this discussion.

4.7.1 SECURITY AGAINST BRUTE-FORCE ATTACKS

The most obvious and most important difference is that the SHA-1 and RIPEMD-160 digest is 32 bits longer than the MD5 digest. Using a brute-force technique, the difficulty of producing any message, having a given digest, is on the order of 2^{128} operations for MD5 and 2^{160} for SHA-1 and RIPEMD-160. Again, using brute-force technique, the difficulty of producing two messages having the same message digest is in the order of 2^{64} operations for MD5 and 2^{80} for SHA-1 and RIPEMD-160. Thus, SHA-1 and RIPEMD-160 are considerably stronger against brute-force attacks. All three algorithms are invulnerable to attacks against weak collision resistance. At 128-bits, MD5 is highly vulnerable to a birthday attack[2] on strong collision resistance, whereas both SHA-1 and RIPEMD-160 are safe for the foreseeable future.

4.7.2 SECURITY AGAINST CRYPTANALYSIS

MD5 is vulnerable to cryptanalytic, while SHA-1 appears not to be vulnerable to such attacks. Little is known about the design criteria for SHA-1, so its strength is more difficult to judge, but it appears to be highly resistant to known cryptanalytic attacks. RIPEMD-160 was designed specifically to resist known cryptanalytic attacks. The use of two lines of processing, which doubles the number of steps performed, gives RIPEMD-160 added complexity, and should make RIPEMD-160 more secure against cryptanalytic attacks compared to SHA-1.

4.7.3 SPEED

Because all algorithms rely heavily on addition modulo 2^{32}, both do well on a 32-bit architecture. SHA-1 and RIPEMD-160 involves more steps (80/160 versus MD5's 64) and must process a 160-bit buffer compared to MD5's 128-bit buffer. Thus, SHA-1 should execute more slowly than MD5, but faster than RIPEMD-160 at the same

[2] A birthday attack is a name used to refer to a class of brute-force attacks.

hardware. Table 4.1 shows a set of results achieved on a 266-MHz Pentium and a Celeron 850-MHz.

Algorithm	Mbps (266 MHz)	Mbps (850 MHz)
MD5	32,4	805,9
SHA-1	14,4	387,7
RIPEMD-160	13,6	245,8

Table 4.1 – A speed comparison of MD5, SHA-1 and RIPEMD-160
Relative performance of the hash functions. Column two is the result on a 266-MHz Pentium and column three is the result on a 850 MHz Celeron [Dai00].

4.7.4 SIMPLICITY AND COMPACTNESS

All algorithms are simple to describe and simple to implement and do not require large programs or substitution tables.

4.7.5 LITTLE-ENDIAN VERSUS BIG-ENDIAN ARCHITECTURE

MD5 and RIPEMD-160 use a little-endian scheme for interpreting a message as a sequence of 32-bit words, whereas SHA-1 uses a big-endian scheme. There appears to be no strong advantage to either approach.

4.7.6 TABLE OF COMPARISON

	MD5	SHA-1	RIPEMD-160
Digest length	128 bits	160 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	64 (4 rounds of 16)	80 (4 rounds of 20)	160 (5 paired rounds of 16)
Maximum message size	∞	$2^{64} - 1$ bits	$2^{64} - 1$ bits
Primitive logical functions	4	4	5
Additive constants used	64	4	9
Endianness	Little-endian	Big-endian	Little-endian

Table 4.2 – A comparison of MD5, SHA-1 and RIPEMD-160
This table highlights some of the differences and similarities between the algorithms [Sta98].

4.8 SELECTING A HASH ALGORITHM

There are many hash algorithms available, but MD5, SHA-1 and RIPEMD-160 are all well known and widely used. MD5 has some flaws, but both SHA-1 and RIPEMD-160 are safe for the foreseeable future. Of these algorithms, SHA-1 is the only one that is FIPS approved by NIST [Bou03], meaning that the algorithm is approved for the use in governmental agencies in the United States.

We considered the SHA-1 and RIPEMD-160 algorithms to be equally secure, but we ended up selecting the SHA-1 algorithm for our new system prototype, because it is easiest to implement using standard Java classes.

4.9 IBUTTON TECHNOLOGY

When developing an authentication mechanism there are many technologies on the market. The Magnetic Stripe cards and Smart cards are well known examples. Dallas Semiconductor / MAXIM have developed another alternative called iButton. The FieldCare prototype is already using iButtons as MiTags. Since iButtons also can be used as authentication devices for personnel, it was natural to further investigate this technology. This section will introduce the iButton technology, while in section 4.10 we present different iButton types that can be used in our new system prototype. Section 4.11 will review the security with iButtons.

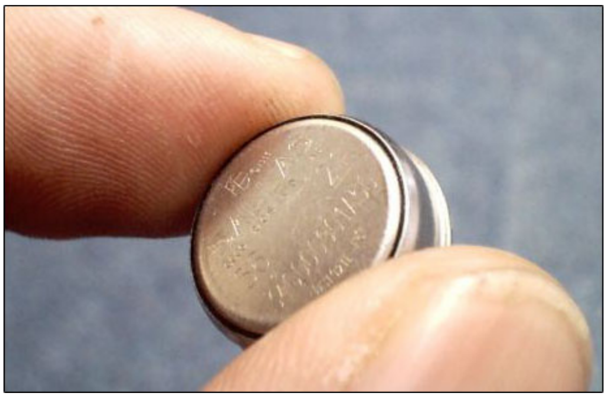


Figure 4.1 – An iButton
This picture shows the size of the iButton in relation to a human finger.

The iButton is a computer chip enclosed in a 16mm stainless steel can. Because of this unique and durable stainless steel can, up-to-date information can travel with a person or object anywhere they go. The steel button can be mounted virtually anywhere because it is rugged enough to withstand harsh environments, indoors or outdoors. The iButton can be dropped, stepped on and scratched, and still work. It is durable enough to attach to a key fob, ring, watch, or other personal items and used daily for applications such as access control to buildings and computers, and it is wear-tested for 10-year durability [DSC97].

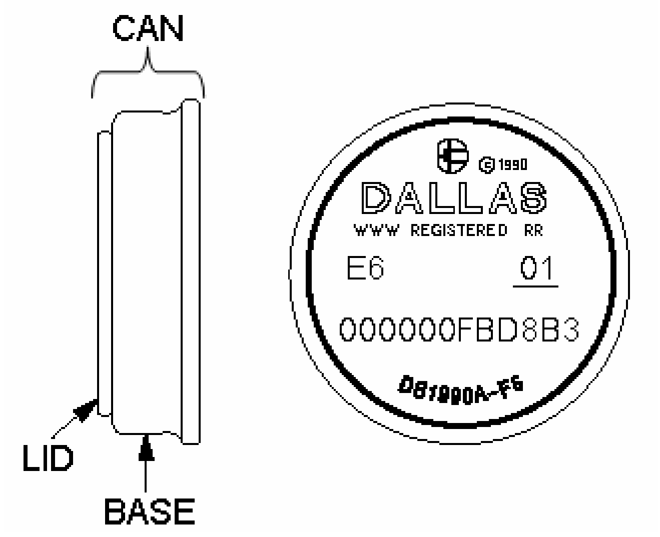


Figure 4.2 – A schematic view of an iButton

All iButtons use their stainless steel *Can* for their electronic communications interface. Each Can has a data contact which is called the *Lid* and a ground contact which is called the *Base*. Each of these contacts is connected to the silicon chip inside. The Lid is the top of the Can and the Base forms the sides and the bottom of the Can and includes a flange for easily attaching the button to just about anything. Each iButton has a unique and unalterable identification number that is laser etched onto the chip inside the can. This address can be used as a key or identifier for each iButton.

By simply touching each of the two contacts you can communicate to any of the iButtons by using the 1-Wire protocol. The protocol, designed by Dallas Semiconductor, is a strictly defined serial protocol. The transfer with iButton is done half-duplex (i.e. either transmit or receive). The 1-Wire protocol is based on a master/slave configuration, in which the host PC or microprocessor is the master and the iButton device is the slave. Multiple iButtons are capable of being on the 1-Wire bus at any given time. The 1-Wire interface has two communication speeds; standard mode at 16 Kbps and overdrive mode at 142 Kbps. Information is transferred between an iButton and a PC with a momentary contact. You simply touch your iButton to a Blue Dot Receptor or other iButton probes, which is connected to a PC. The Blue Dot receptor, shown in Figure 4.3, is cabled to a 1-Wire adapter that is attached to the PCs serial or parallel port. All of the latest handheld computers and PDAs can communicate with iButtons [Kin00].

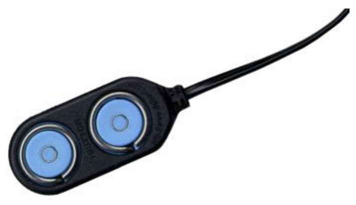


Figure 4.3 – The Blue Dot Receptor
The Blue Dot Receptor used to access iButtons

4.10 IBUTTON TYPES

There are many types of iButtons from Dallas Semi Conductor to choose from. This section gives an overview of iButtons that are suitable for our scenario. All iButtons have a specific family code defined by the iButton's functionality and capacity and a unique serial number that is engraved into the iButton Can and laser etched to the chip inside. This is a 48-bit serial number that can represent any decimal number up to $2.81*10^{14}$. (If 1000 billion ($1.0*10^{12}$) iButtons of the same family code were produced per year, this number range would be sufficient for 281 years. In addition there are 128 different family codes available.)

4.10.1 NV RAM IBUTTONS

These iButtons contains Non-Volatile (NV) Random Access Memory (RAM), and they can function as an identification tag and a data carrier at the same time. The serial number can be used as a seed together with a secret key to encrypt non-public data files. Table 4.3 shows an overview of the different NV RAM iButton types.

Device Type	Memory Size
DS1992	1Kb
DS1993	4Kb
DS1995	16Kb
DS1996	64Kb

Table 4.3 – An overview of the different NV RAM iButtons
The NV RAM iButtons contain various memory sizes;
the smallest with 1Kb and the largest with 64Kb

4.10.2 EEPROM IBUTTONS

There are two Electrically Erasable Programmable Read-Only Memory (EEPROM) iButtons; DS1971 and DS1973. The DS1971 contains a one-page 256-bit EEPROM block. It also contains a 64-bit one-time programmable application register. The DS1973 has a larger EEPROM (4Kb) and a 256-bit scratchpad (temporary) memory, but it doesn't have the one-time programmable register.

4.10.3 PASSWORD PROTECTED MEMORY IBUTTONS

This model is called DS1991 and is tamper-proof. It contains three blocks of 384-bit memory that each can be password protected with 64-bit passwords and a 512-bit scratchpad memory. If not the correct password is used to read data, the iButton will output random numbers. You have the opportunity to use the scratchpad memory as general-purpose memory, but it is then unprotected.

4.10.4 CRYPTOGRAPHIC IBUTTONS

There are two cryptographic iButtons; DS1955 and DS1957, and are called Java-powered iButtons. Both have a Java virtual machine (VM) which is Java Card 2.0 compliant[3]. The DS1955 has been approved for the protection of sensitive, unclassified information by the National Institute of Standards (NIST) and the Communications Security Establishment (CSE). These iButtons have their own 1024-bit math processor which performs public key cryptography in less than one second, a garbage collector that collects any objects that are out of scope and recycles the memory for future use, and a non-volatile memory up to 136 Kb. The chip also includes up to 136 Kb SRAM which is specially designed so that it will rapidly erase its contents as a tamper response to an intrusion.

4.11 SECURITY WITH IBUTTONS

Many of the attacks that the iButton is vulnerable to are common to other embedded system attacks. [KK99] have defined four high-level attack categories which were used in smartcard analysis and could be applicable to the iButton [Kin00].

[3] A specification enabling Java technology to run on smart cards and other devices with limited memory [Sun03a].

- Micro probing consists of invasive techniques used to access the device internals directly.

- Software attacks rely on the normal communications interface of the device and, by having the device execute custom software, exploit security vulnerabilities.

- Eavesdropping techniques monitor the external connections to the device and any stray Electromagnetic Interference (EMI)/Radio Frequency (RF) noise radiated from the device during normal operation.

- Fault generation techniques use abnormal environmental conditions to intentionally cause failures in the device. By doing so, the device may function outside of its intended feature set.

Differential Power Analysis[4] cannot readily be used, unless invasive methods are successful in accessing the iButton's internal battery power supply. However, it might be useful to analyze the current power consumption and the characteristics of the 1-Wire interface.

4.11.1 INVASIVE ATTACKS

Invasive attacks require access to the physical device and are often destructive. Depending on the complexity of the device, special laboratory techniques and chemicals might be used. An example would be the de-packaging of smartcards [KK99], in which hot fuming nitric acid is used to gain access to the die.

Many devices contain tamper proofing mechanisms against invasive attacks, which will render the device inoperable if tampering is detected. In order for invasive attacks to be successful, access to the devices internals must be completed without detection.

4.11.1.1 UNIQUENESS

The 64-bit identification, comprising the family code, serial number, and Cyclic Redundancy Code (CRC) is guaranteed by Dallas Semiconductor to be unique. Systems designed solely around uniqueness often have problems. An example of this is the Ethernet Media Access Control (MAC) addresses.

Ethernet MAC addresses are fairly easy to clone in both hardware and software, allowing one to bypass copyright protection and launch denial-of-service and race condition attacks.

It might be possible to clone the iButton by modifying the unique ID by invasive or non-invasive techniques. If the 48-bit unique serial number is changed, the CRC must also be changed, since the value will now be incorrect. The 64-bit unique ID is marked onto the iButton in two different locations:

[4] This attack is performed by monitoring the electrical activity of a device, and then using advanced statistical methods to determine secret information [KJJ03].

- *Stainless Steel Housing.* By etching the housing, this ID could easily be altered.

- *Internal Read Only Memory (ROM).* The ID is created by selectively removing 3-micron polysilicon links, each which define a *zero* or *one*. The links are sealed with a protective layer of glass, so that any tampering attempts would be evident. It may be possible to vary the ID by opening closed links or reforming open links. If the device still functions and the particular attack is successful, having physical evidence of tamper proofing, such as the broken glass, is a debatable point [Kin99].

4.11.1.2 NV RAM ACCESS

A common attack, fitting into the *micro probing* category, is to gain physical access to the PIN connections or wire bounds of the internal memory of a device and dump all memory locations. This is done in the hope of obtaining critical data, encryption keys or other information that is considered to be secure.

The Java-Powered Cryptographic iButton includes a tamper-detection feature that will immediately erase all internal NV RAM when the physical perimeter of the device is compromised.

4.11.1.3 PROCESSOR CLOCK SKEWING

Gaining external control of the clock signal of the system can be a valuable tool.

- *Increasing clock speed.* This will allow the attacker to view more iteration of calculations or look for repeated sequences. This attack is useful when analyzing time-based hardware token devices.

- *Decreasing clock speed.* This will allow the attacker to slow down or single-step through operations and analyze the data using external measurement tools.

- *Halting system execution.* Doing so at a known point will allow the attacker to repeatedly examine a particular operation condition more closely. This is done by applying the same number of clock cycles each time after reset.

- *Inducing calculation errors.* This is often possible by inducing abnormal clock signals into the system. Calculation or execution errors may lead to conditions where critical information is leaked or incorrectly handled.

The processor internal to the Java-Powered Cryptographic iButton is driven by an un-stabilized oscillator ranging from 10 to 20 MHz. By having the iButton vary its clock frequency, it makes clock skewing attacks more difficult.

4.11.2 NON-INVASIVE AND SOFTWARE ATTACKS

With non-invasive attacks, the physical device is not harmed or tampered with. Non-invasive and software attacks often, but not always, make use of the normal operation conditions of the device. Once the attack is designed and successful, the results are extremely reproducible from one device to another.

Due to the fact that communication is only done through the 1-Wire interface, many of the software attacks on the Java-Powered Cryptographic iButton are launched through this interface. A Java program would be loaded into the device using the normal communications means, and executed normally as a Java applet. The possibility of software attacks are great and one should not think easy on this area. There are specific Java related problems that could be used to help launch an attack, including byte code verification, problems with code signing, and resource starvation. Any Java applet loaded onto the Java-Powered Cryptographic iButton can be a target for attacks [Kin99].

4.11.2.1 SPOOFING

In the same vein of the invasive uniqueness attacks, a non-invasive method of spoofing of the 1-Wire communication signal is entirely feasible. By monitoring the communication stream of the iButtons during normal usage, the secret key or other critical information may unknowingly be transmitted.

Imitating an iButton with electronic circuitry to clone a device would also be possible. Full device emulation would be easier accomplished with the low featured iButtons (RAM iButtons). Depending on the system implementation, it would be trivial to imitate an iButton and transmit its cloned 64-bit unique ID to the host PC in hopes of the host transmitting and revealing critical data.

4.11.2.2 MISUSE OF MEMORY MANAGEMENT

If hardware or software memory management is unused, misused or not configured properly, it may be possible to access protected memory and scratchpad areas that are considered critical. Due to programming errors of particular Java applications running on the iButton, critical data or secret keys might not be cleared from memory, thus giving the attacker information that may be used in future attacks. The attacking software application would attempt to dump all memory areas and access any memory-mapped peripherals. The old Kerberos 4 exploit took advantage of such a situation, in which the username and Kerberos realm of the previously logged in user was recovered due to improper clearing of critical memory [Zat96].

4.12 DISCUSSION

There are, as presented in section 4.3, many different types of access control. It is very difficult to define one access control to FieldCare. FieldCare controls access to the device and application through a login procedure. The login procedure in the FieldCare prototype, was very static and has been changed by us. The new login procedure, explained in section 4.1, implements an access control, by using PiTags containing user information and a hashed PIN code for authentication.

Our new system prototype uses either *User* or *Emergency* login, thus you could say this is *Role Based Access Control*. The *User* login could be thought of as a *User Based Access Control*, due to the fact that the users to a certain degree get different access. I.e. all users have the ability to insert data, but only the authenticated users get access to retrieving information from the hospital, and the amount of data they receive depend on what access level (i.e. Competence Role) they have. Hence you can say that the retrieval of patient journals use a *View Based Access Control*.

As we mentioned in section 4.8, there are many hash functions available. Some of the widely used algorithms are MD5, SHA-1 and RIPEMD-160. While there are some weaknesses with MD5, both SHA-1 and RIPEMD-160 are considered secure. The standard Java Application Programming Interface (API) has built in hash functions supporting the SHA-1 algorithm, but RIPEMD-160 is not supported in this API.

In our implementation we use the iButton as PiTag. This choice was made because SINTEF Telecom and Informatics already had used the iButton as MiTag. There are many iButtons to choose from, as we mentioned in section 4.10. They are all very robust and durable, two properties that are of great importance in the outdoor environment of the FieldCare WLAN.

A cryptographic java-powered iButton could have been interesting in our implementation, but since we only are going to store very little information on the iButton and this project is just a prototype, we are simply using a NV RAM iButton. To secure the vital information for the authentication, we are making a message digest of the PIN code, by using a hash algorithm.

The solutions presented in this chapter, e.g. using the iButton as PiTag and combining the use of a PiTag with a hashed PIN code for authentication, satisfy the following security requirements as specified in 3.4:

- Securing MiTag and PiTag Data (Partly satisfied when hashing the PIN code)
- Durability (Partly satisfied with the use of iButtons)
- User Authentication
- Emergency Login Restriction

4.13 SUMMARY

We have looked at how we can improve the existing access control in the FieldCare prototype. By choosing iButtons for use as PiTags, which SINTEF Telecom and Informatics already are using as MiTags, together with a hashed PIN, makes the authentication mechanism stronger. Storing the PIN code in clear-text would give us no security if the iButton should be stolen. We will use the SHA-1 algorithm to hash the PIN code, because this algorithm is secure and easy to implement. We have also introduced a new concept, called Competence Role. When a user is authenticated using a PiTag and the PIN code, a Competence Role will grant the user access to download Patient Journals.

5 DYNAMIC ROUTING

This chapter will provide an overview of some of the current proposals and implementations of dynamic routing protocols for wireless mobile ad-hoc networks. The first section is a brief introduction to ad-hoc networking. Section 5.2 presents the various routing protocols, while section 5.3 will focus on the security aspects associated with this type of protocols. In 5.4 we compare and test some of the protocols from 5.2 and we select one protocol to incorporate in the FieldCare application.

5.1 INTRODUCTION

A mobile ad-hoc network (MANET) is a system of wireless mobile nodes that are dynamically self-organizing in arbitrary and temporary network topologies [BCG01]. In other words, it is an infrastructureless peer-to-peer (P2P) network formed by a set of nodes within the range of each other, where all nodes acts as routers and take part in the discovery and maintenance of routes to the other nodes in the network.

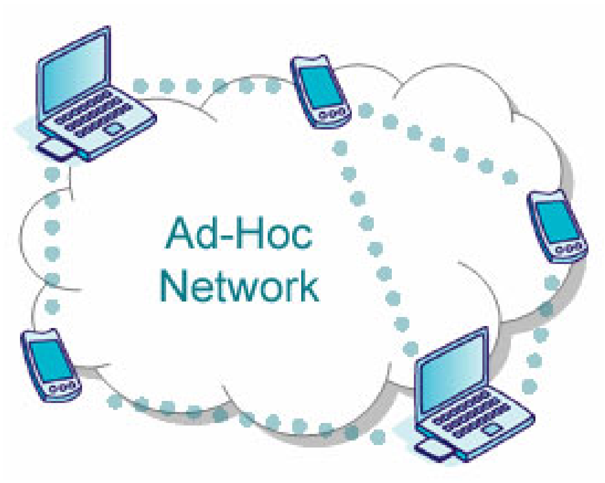


Figure 5.1 – Example of an ad-hoc network topology
The figure shows an example of an ad-hoc network formed between five nodes.

Ad-hoc networks are very useful in emergency search-and-rescue operations, data acquisition operations in inhospitable terrain and meetings or conventions in which people may want to quickly share information [Roy99]. After the attack on the World Trade Center on September 11th 2001, most of the communication was paralyzed in the surrounding area due to fact that the base stations covering the area had been installed on the rooftops of the collapsed buildings. Rescue operations in combination with large disasters, terror attacks etc. have a need for wireless networks that are independent of any existing infrastructure. Such a network would ease the coordination of the rescue operation by quickly distributing information to the right people and it makes it possible to deploy sensors for remotely observing the condition of patients. One could also establish a communication link to a nearby hospital and provide them with information before the patients arrive [VGE03]. All this is very much applicable to the FieldCare project.

The traditional routing protocols are made for networks that are interconnected with wires. These protocols use too great a time to adapt to the rapid topology changes in a MANET, and they also involve too much routing traffic. The new routing protocols suited for MANETs are either optimized versions of the traditional protocols or completely designed from scratch. In general, the MANET routing protocols are grouped in two categories; *proactive* and *reactive*.

A proactive routing protocol maintains all the routes at all time, independent of whether or not the link is being used, and since the route is pre-computed, data may be transmitted instantly. Proactive protocols are typically optimized versions of traditional routing protocols for wired networks.

A reactive routing protocol only computes a route when needed. This results in a highly optimized link utilization, as no unnecessary traffic is being generated. However, the disadvantage is the time delay (when computing the route) prior to a data transmission.

5.2 DYNAMIC ROUTING PROTOCOLS

This section will present different proposals and implementations of dynamic routing protocols for mobile ad-hoc networks. A great number of protocols have been suggested (CEDAR, ABR, TORA, ZRP, just to mention a few of them), but we will only focus on those that are advancing within the Internet Engineering Task Force (IETF). Please note that we will not examine the technical details of each protocol as that is not part of this thesis.

5.2.1 OPTIMIZED LINK STATE ROUTING

The current specification of the Optimized Link State Routing (OLSR) protocol is defined in [CJ03] as an optimization of the classical link state algorithm tailored to the requirements of a mobile WLAN. OLSR is a table driven and proactive protocol that applies a multi-tiered approach with *multi-point relays* (MPR). In OLSR, topology information is exchanged regularly with other nodes of the network. The nodes selected as MPRs by some neighbor nodes, dispense topology information periodically in their control messages. This technique substantially reduces the message overhead as compared to a classical flooding mechanism, where every node retransmits each message when it receives the first copy of the message. The link state information is generated only by the nodes elected as MPRs. Thus, a second optimization is achieved by minimizing the number of control messages flooded in the network. And finally, as a third optimization, a MPR node may choose to report only the links between itself and its MPR selectors. Hence, as contrary to the classic link state algorithm, only partial link state information is distributed in the network [CJ03].

OLSR operates in a totally distributed manner, e.g. the MPR approach does not require the use of centralized resources. Also, the MPR approach is most suited for large and dense ad-hoc network, where traffic patterns are random and sporadic. Hence the OLSR protocol works best in this kind of environment.

The MPRs are chosen so that only nodes with one-hop symmetric (bi-directional) link to another node can provide the service. Thus in very dynamic networks where it constantly exists a substantial amount of unidirectional links, the MPR approach used in OLSR may not work properly [Kar00].

The performance of the OLSR protocol as compared to reactive protocols is increased if the source-destination pairs change with time. Such changes may trigger substantial query flooding in the case of a reactive protocol but not in OLSR as the routes are maintained for each known destination, at all times [SB02].

5.2.2 AD-HOC ON-DEMAND DISTANCE VECTOR

The Ad-hoc On-demand Distance Vector (AODV) protocol is a unicast-based reactive routing protocol specifically designed for mobile nodes in an ad-hoc network. It offers quick adaptation to dynamic link conditions, low processing and memory overhead, low network utilization, and determines unicast routes to destinations within the ad hoc network [PBD03]. Since AODV is a reactive protocol, the nodes in the network maintain the topology dynamically only when there is traffic, hence minimizing the number of required broadcasts.

The authors of AODV classify it as *a pure on-demand route acquisition* system, since nodes that are not on a selected path do not maintain routing information or participate in routing table exchanges [Roy99].

5.2.3 TOPOLOGY DISSEMINATION BASED ON REVERSE-PATH FORWARDING

The Topology Dissemination Based on Reverse-Path Forwarding (TBRPF) protocol is a proactive, link-state routing protocol designed for mobile ad-hoc networks, which provides a hop-by-hop routing along shortest paths to each destination. Each node that uses TBRPF computes a source tree (i.e. providing paths to all reachable nodes) based on partial topology information stored in its topology table. To minimize the overhead, each node will only report a part of its source tree to the neighbors. TBRPF uses a combination of periodic and differential updates to keep all neighbors informed of the reported part of its source tree [OLT03].

TBRPF can support networks with up to a few hundred nodes, and can be combined with hierarchical routing techniques to support much larger networks. It is particularly well suited to MANETs consisting of mobile nodes with wireless network interfaces operating in P2P fashion over a multiple access communications channels [OLT03].

5.2.4 DYNAMIC SOURCE ROUTING

The Dynamic Source Routing protocol (DSR) is a simple and efficient reactive routing protocol designed specifically for use in multi-hop wireless ad-hoc networks of mobile nodes [JMH03]. It uses source-routed on-demand routing where each node maintains a route cache containing the source routes that it is aware of. This cache is updated with better routes, if existent, when it discovers new routes and hosts.

DSR eliminates the need for every node in the network to do periodic route discovery advertisements by requiring that each packet keeps its route information. DSR allows the network to be completely self-organizing and self-configuring, without the need for any existing network infrastructure or administration. DSR does not make any periodic update messaging, thus avoiding wasting more bandwidth than necessary.

The DSR protocol is designed mainly for mobile ad hoc networks of up to about two hundred nodes, and is designed to work well with even very high rates of mobility [JMH03].

5.3 SECURITY IN AD-HOC NETWORKS

Ad-hoc networks have characteristics that are quite different from conventional wired networks. E.g. the lack of a preexisting infrastructure introduces problems in routing, name resolution, service discovery and of course in security. The security aspects that are valid in the networks of the past are not necessarily applicable in ad-hoc networks. While basic security requirements such as confidentiality, integrity and authenticity remain, the ad-hoc networking approach somewhat restricts the set of feasible security mechanisms to be used, as the level of security and performance are related to each other. Currently the amount of available memory and processing power is relatively small; making the implementation of strong protections for ad-hoc networks a non-trivial-problem [Kar00]. Achieving secure communication is especially challenging in ad hoc networks, where the taxonomy is server-less, and the state and structure of the network is dynamic and fault-prone

Mobile ad-hoc network protocols are now being designed without security in mind. Most of the protocol specifications assume that all the nodes in the network are friendly. The security issue has been postponed and there exists a common feeling that it is possible to secure the routing protocols by simply retrofitting preexisting cryptosystems. But securing the network transmission without securing the routing protocol is not sufficient [Zap02].

The FieldCare WLAN needs to be secured in accordance with the security specification provided in section 3.4. We will therefore first examine various possible attacks against WLANs, and then review different approaches for securing both the routing protocol used and the data flow.

5.3.1 ATTACKS

An ad-hoc network's ability to securely distribute routing data is critical to the functioning of the network and the network must protect the routing data from malicious attacks. These attacks can be categorized into two groups; *passive* and *active* attacks. And the routing control data is susceptible to both types of attack [TL01]. Each group can also be subcategorized into *external* and *internal* attacks.

Passive attacks are recognized by disclosing information without changing the state of the system being monitored, while active attacks are a deliberate unauthorized change to a state of a system. The following subsections present various attacks from each category.

5.3.1.1 EAVESDROPPING (PASSIVE AND EXTERNAL/INTERNAL)

According to [T1A101] the definition of eavesdropping is; *the unauthorized interception of information-bearing emanations*. Any communication must be protected from eavesdropping, whenever confidential or sensitive information is exchanged. Also critical data the nodes store must be protected from unauthorized access. Control data is sometimes more critical information, with respect to the security, than the actual data being exchanged. E.g. the routing directives in packet headers, such as the identity or location of the nodes, could in a military scenario be more valuable to the enemy than the application-level messages.

5.3.1.2 TRAFFIC ANALYSIS (PASSIVE AND EXTERNAL/INTERNAL)

The definition of traffic analysis is; *the inference of information from observation and analysis of the presence, absence, amount, direction and frequency of the traffic flow* [T1A101]. Using traffic analysis, an attacker may gain valuable information about the data flow, and maybe with sufficient time[5] be able to break the cryptographic scheme used in the network.

5.3.1.3 DENIAL OF SERVICE (ACTIVE AND EXTERNAL)

A denial of service (DoS) attack is defined by [T1A101] as; *the prevention of authorized access to resources or the delaying of time-critical operations. The result of any action or series of actions that prevents any part of an information system from functioning*. Compromised nodes may be able to reconfigure the routing protocol or any part of it so that they send routing information very frequently, thus causing congestion or preventing the nodes to gain new information about the changed the topology of the network. In the worst cast the attacker is able to change the routing protocol to operate arbitrarily or perhaps even in the (invalid) way the attacker wants. If the compromised nodes and the changes to the routing protocol are not detected, the consequences are severe, as from the viewpoint of the node the network may seem to operate normally. This kind of invalid operation of the network initiated by malicious nodes is called a *Byzantine failure* [Kar00]. A special type of DoS attacks is called distributed denial of service (DDoS) attacks and involve a greater number of a nodes controlled by the attacker.

5.3.1.4 JAMMING (ACTIVE AND EXTERNAL)

[T1A101] defines jamming as; *the deliberate radiation, re-radiation or reflection of electromagnetic energy for the purpose of disrupting enemy use of electronic devices, equipment or systems*. If an attacker has a powerful transmitter, he or she can generate a radio signal strong enough to overwhelm weaker signals, disrupting the communication. Jamming attacks could also be defined as a form of DoS attack. Two types of jammers that may be used against WLAN traffic are high-power pulsed full-band jammers that cover the entire frequency used by the targeted signal, and lower-power partial-band jammers that cover only part of the frequency used by the targeted signal. Jamming is successful when the jamming signal denies the usability of the communications transmission. In digital communications, the usability is denied when

[5] Not applicable to the WLAN network used in FieldCare, as this is a highly temporal network that will not provide the attacker with sufficient time. However, it is given that the network uses a strong crypto scheme, as 128-bit WEP keys may be broken in about 15 minutes [FMS02].

the error rate of the transmission cannot be compensated by error correction [Sta00]. Jamming equipment is readily available to consumers or can be constructed by knowledgeable attackers.

5.3.1.5 IMPERSONATION (ACTIVE AND INTERNAL)

Impersonating is defined by [T1A101] as a form of spoofing, which in turn is defined as; *the interception, alteration and retransmission of a cipher, signal or data in such a way to mislead the recipient*. If proper authentication of parties is not supported, compromised nodes may at the network layer be able to e.g. join the network undetectably or send false routing information masqueraded as some other, trusted node. The attacker could gain high access privileges.

5.3.1.6 BLACK HOLE (ACTIVE AND INTERNAL)

A black hole is generated when a compromised or a malicious node advertises the shortest path to a destination node, and is thereby used as a relay node to reach the destination. The compromised or malicious node then creates a void or a black hole by intercepting all received packets without forwarding them.

5.3.1.7 WORMHOLE ATTACK (ACTIVE AND EXTERNAL/INTERNAL)

In a wormhole attack, an attacker records packets (or bits) at one location in the network, tunnels them (possibly selectively) to another location, and retransmits them there into the network. The wormhole attack is possible even if the attacker has not compromised any hosts and even if all communication provides authenticity and confidentiality. The wormhole attack can form a serious threat in wireless networks, especially against many ad-hoc network routing protocols. A wormhole attack would result in the routing protocol being unable to find routes longer than one or two hops, severely disrupting communication. All the routing protocols presented in 5.2 are vulnerable to the wormhole attack. [HPJ03] presents a defense against wormhole attacks by using a new mechanism they call *packet leaches*, where information is added to a packet designed to restrict the packet's maximum allowed transmission distance or lifetime. The use of this mechanism makes the receiver able to detect and possibly prevent wormhole attacks.

5.3.2 ROUTING PROTOCOL SECURITY CONSIDERATIONS

As stated in 5.3, most of the work concerning the security in wireless networks are focused on securing the transmission or flow of data, while there may exists several vulnerabilities in the routing protocol itself. The following subsections will examine the security in the ad-hoc routing protocols presented in 5.2.

5.3.2.1 SECURITY IN OLSR

The Internet Draft document [CJ03] specifying the OLSR protocol states that no special security measures have yet been taken into consideration. However the document does specify possible vulnerabilities. Since OLSR is a proactive protocol, it periodically disperses topological information, and this information is revealed to anyone listening to the OLSR control messages. The network integrity may be compromised if some malicious nodes inject invalid control traffic into the network, and is why [CJ03] recommends using authentication of the control messages and temporal information that allows a node to positively identify delayed messages. However the document does not say anything on whether or not these recommendations will be implemented in future OLSR versions.

5.3.2.2 SECURITY IN AODV

No security measures are specified in the AODV draft [PDB03], however the authors identify the necessity of having proper confidentiality and authentication mechanisms within the routing. Interestingly the Nokia Research Center has developed a specification for an improved AODV protocol called the Secure Ad hoc On-Demand Distance Vector Routing (SAODV) protocol [Zap01]. The SAODV protocol is an extension of the AODV protocol that can be used to protect the route discovery mechanism by providing security features like integrity, authentication and non-repudiation of the routing messages.

5.3.2.3 SECURITY IN TBRPF

There are no security mechanisms specified in the TBRPF specification either [OLT03], and again the document simply states that wireless networks are vulnerable to a variety of attacks. However the authors of TBRPF are a bit more specific regarding possible security solutions. They suggest that TBRPF packets can be authenticated by using the IP Authentication Header, which is a part of the IP Security (IPSec) architecture. In addition, the Encapsulating Security Payload (ESP) header, also a part of IPSec, can be used to provide the confidentiality of TBRPF packets.

5.3.2.4 SECURITY IN DSR

It should not come as big surprise that the specification document for the DSR protocol [JMH03] has not implemented any security mechanisms either, as none of the other protocol specifications presented here has done so. But the level of naivety in this document is greater than in the other specifications, as the authors assume that all the nodes participating in the DSR protocol do so in good faith and without malicious intent to corrupt the routing ability of the network. No security recommendations are made whatsoever, except the suggestion of using encryption at the link layer.

5.3.3 DATA FLOW SECURITY CONSIDERATIONS

Securing the data flow has usually been given more attention than securing the routing protocol, and hence there exists more available options. However, as subsection 5.3.3.1 below shows, some security implementations are questionable.

5.3.3.1 WEP

The current standard for protecting wireless communication from eavesdropping is the Wired Equivalent Privacy (WEP) algorithm. A secondary function of the WEP algorithm is to prevent unauthorized access to a wireless network.

WEP is a symmetric cryptosystem, which relies on a secret key being shared between the nodes in the network. The secret key is used to encrypt packets before they are transmitted, and an integrity check is used to ensure that packets are not modified in transit. The WEP standard does not specify any techniques for key distribution, but in practice, most installations use a single key that is manually shared between all nodes [BGD01].

In Appendix A.7 we provide a configuration of WEP on the devices used in our setting (MDA, CMDA and FCC). We enabled 128-bit WEP on all the WLAN devices, but we recommend using an alternative cryptosystem when deploying FieldCare. WEP's insecurities are discussed in detail in the next paragraph. WEP was enabled because it provided a link-to-link encryption that was non-existent in the previous prototype of FieldCare, though insecure, it's is better than transmitting in plaintext and it is now more in compliance with the security specification from section 3.4.

There are many security problems associated with the WEP algorithm, as some researchers from the University of California at Berkeley discovered [BGD01]. WEP uses the RC4 encryption algorithm, which is the most widely used stream cipher in software applications [FMS02]. A stream cipher operates by expanding a short key into an infinite pseudo-random key stream. The sender exclusive or (XOR) the key stream with the plaintext to produce a ciphertext. The receiver, which has a copy of the same key, uses this key to generate an identical key stream and XORs this key stream with the ciphertext to produce the plaintext. This mode of operation renders the stream cipher vulnerable to several attacks, e.g. if the attacks flips a bit in the ciphertext this error is carried over to the plaintext, and if the attack intercepts two ciphertexts encrypted with the same key stream, the plaintext can be obtained by XORing the two ciphertexts. WEP has implemented defenses against these kinds of attacks, in the form of an Integrity Check (IC) that ensures the packet has not been modified in transit and an Initialization Vector (IV) that is used to augment the shared key and produce a different RC4 key for each packet. The problem is that these measures are poorly implemented [BDG01].

The IC is implemented as a CRC-32 checksum, and since this CRC is linear it means that it is possible to compute the bit difference of two CRCs based on the bit difference of the messages over which the checksums are generated. This allows the attacker to flip arbitrary bits in an encrypted message and correctly adjust the checksum so that the resulting message appears valid to receiver. The IV in WEP is only 24-bit, and with such a small space of possible initialization vectors, the reuse of the same key stream is inevitable. Testing reveals that WEP encapsulation remains insecure whether its key length is 1 or 1000 bits, and it is not due to the use of the RC4 algorithm but rather WEP's incorrect usage of an IV [Wil01].

5.3.3.2 LEAP

As an alternative to WEP, Cisco Systems has devised a security scheme called LEAP (Lightweight Extensible Authentication Protocol) which is based on the Extensible Authentication Protocol (EAP) [BVA^{+}03], an integral part of the 802.1x authentication framework. LEAP mitigates several of the weaknesses in WEP by utilizing a dynamic WEP key and sophisticated key management. The dynamic 128-bit WEP key used by LEAP is changing on a per-user per-session basis for increased security. The LEAP implementation has also improved the usage of an initialization vector, where the IV is now changed on a per-packet basis, creating a non-deterministic sequence. LEAP is currently only implemented for managed wireless networks (i.e. using access points), as key management is not completely solved for the MANET's decentralized structure. LEAP is also currently a highly proprietary solution that only works with Cisco's equipment.

5.3.3.3 IEEE 802.11I

The IEEE 802.11i is a supplemental draft standard currently under development that is intended to improve the WLAN security. Among other things defined in the 802.11i specification are some new encryption key protocols including the Temporal Key Integrity Protocol (TKIP) and the Advanced Encryption Standard (AES) [Eat02]. The 802.11i standard will correct the security problems associated with WEP, e.g. TKIP uses an extended 48-bit IV, compared to the 24-bit IV in WEP. It would take approximately 100 years for a key to be reused when using a 48-bit IV under heavy traffic conditions [Eat02].

5.4 SELECTING A ROUTING PROTOCOL FOR FIELDCARE

All FieldCare computer units (i.e. MDA, CMDA and FCC) currently use IEEE 802.11b network interfaces, which provide a theoretical maximal bandwidth of 11 Mbps, but the real throughput is lower, usually as much as half the theoretical throughput[6]. This fact must be taken into consideration when selecting a routing protocol, since the protocol itself will generate data traffic. Another important consideration is the time it takes each protocol to adapt to changes in the network topology.

5.4.1 OLSR / AODV COMPARISON

Thales Communications AS together with Applica AS has performed a couple of comparison tests between OLSR and AODV. The results of this experiment were presented in [VGE03]. The tables below display the transmitted routing traffic load from each node in different configurations. Table 5.1 shows the result when using OLSR, while the AODV results are presented in Table 5.2.

Configuration	Node 1	Node 2	Node 3	Node 4	Node 5
N1	608 bps				
N1-N2	668 bps	668 bps			
N1-R1-N2	730 bps	730 bps	1,0 kbps		
N1-R1-R2-N2	730 bps	730 bps	1,3 kbps	1,3 kbps	
N1-R1-R2-R3-N2	740 bps	740 bps	1,5 kbps	1,5 kbps	1,5 kbps

Table 5.1 – OLSR test results
The traffic load generated by OLSR for different node configurations. (N = node and R = relay node)

[6] IEEE 802.11b systems running at 11 Mbps use *Complementary Code Keying* (CCK) and *Quaternary Phase Shift Keying* (QPSK) modulation and this may cause the throughput to throttle back to 5.5 Mbps by halving the signaling rate as the distances increase, obstacles appear and error performance drops. Even in near ideal situations the actual throughput of an 802.11b system is much less than the raw bandwidth. Physical layer overhead consumes 30%-50% of the bandwidth. So an 802.11b system running at the full rate of 11 Mbps therefore provides a throughput of only 6 Mbps or so, assuming overhead in the range of 40% [Hor02].

Configuration	Node 1	Node 2	Node 3	Node 4	Node 5
N1	374 bps				
N1-N2	374 bps	374 bps			
N1-R1-N2	374 bps	374 bps	374 bps		
N1-R1-R2-N2	374 bps	374 bps	374 bps	374 bps	
N1-R1-R2-R3-N2	374 bps	374 bps	374 bps	374 bps	374 bps

Table 5.2 – AODV test results
The traffic load generated by AODV for different node configurations. (N = node and R = relay node)

As Table 5.1 shows, the traffic load is steadily increasing as the number of nodes increase. This is particularly true for the relay nodes, as these nodes have to transmit and forward MPR information.

AODV on the other hand, as shown in Table 5.2, has a constant traffic load independent of the number of connected nodes, since each node only transmits information about its status.

Another test performed by [VGE03] was to measure the time used by each protocol to adapt to network topology changes. Figure 5.2 displays the configuration of the test network used.

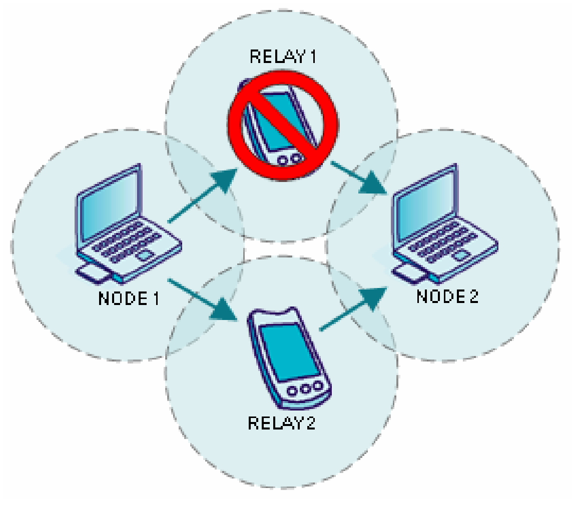


Figure 5.2 – Network for testing convergence time
The network is configured with two parallel relay nodes where the traffic is moved from Relay 1 to Relay 2. The dotted circles do not represent the actual coverage area, but rather that Node 1 has an indirect connection to Node 2 through either Relay 1 or Relay 2.

Data packets were initially transmitted from Node 1 to Node 2 through Relay 1. After a while the WLAN card on Relay 1 was deactivated, forcing the data packets to be transmitted through Relay 2 in order to reach Node 2. The time it took for the protocol to realize that Relay 1 was inactive and to reroute traffic through Relay 2 instead is shown in Table 5.3 below. This value is by most specifications referred to as the *convergence time*.

Protocol	Minimum	Maximum	Average
OLSR	9 s	15 s	12 s
AODV	1,7 s	1,9 s	1,8 s

Table 5.3 – Protocol convergence time
The tables shows the time it takes each protocol to adapt to the new topology shown in Figure 5.2.

The results combined show that the traffic load of the routing is relatively low in both OLSR and AODV, compared to the available bandwidth of the channel, though with OLSR the expected traffic load in a large complex network may be quite significant. The most disturbing result is OLSR's long convergence time. However [VGE03] used in their experiment only default configuration parameters. By changing these parameters the convergence time can be significantly improved [Mac03]. Also the version of the OLSR implementation used in the experiment is an older version (1.0a9), and convergence time is one of many things improved in newer implementations of OLSR.

5.4.2 OUR TEST OF OLSR

OLSR was easy to install on both iPAQs and laptop PCs running Linux, so we decided to run a few tests ourselves and compare our results with those presented in 5.4.1. We installed the same OLSR version as used in [VGE03] (i.e. 1.0a9), and further details about this version are provided in Appendix A.8.

We decided to focus the experiments on OLSR's convergence time, as this value is of great importance in the highly dynamic FieldCare scenario where many nodes will constantly change locations. Our test network had the same basic structure as shown in Figure 5.2. Two iPAQs were used as the relay nodes, while two standard laptops were used as the source and destination nodes. The exact layout of our test network is shown in Figure 5.3 below.

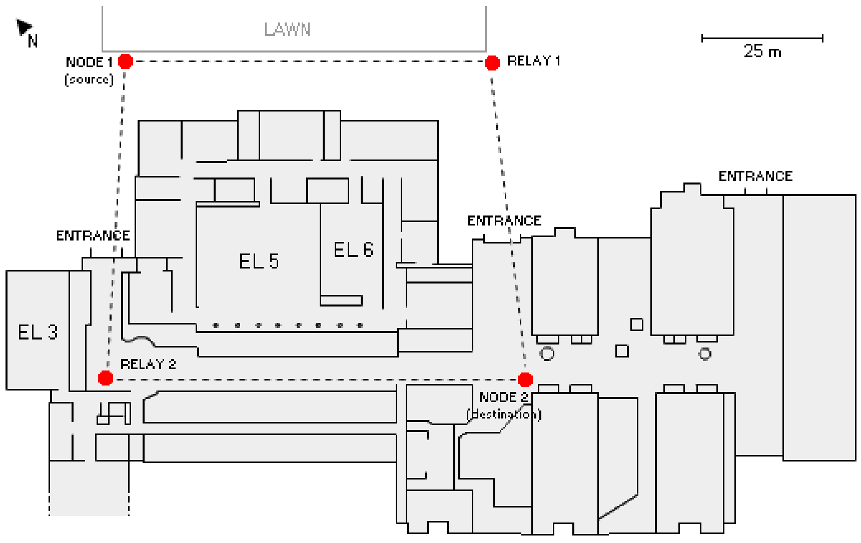


Figure 5.3 – Layout of OLSR test network
The figure shows parts of the Elektrobygget (Department of Telematics) and the location of each node.

The MGEN application [AG03] was installed on Node 1. This application provides the ability to perform IP network performance tests and measurements using User Datagram Protocol (UDP) traffic. The application generates real-time traffic patterns so that the network can be loaded in a variety of ways. The generated traffic can also be received and logged for analysis by the DREC tool installed on Node 2. Appendix B.1 contains more details about these applications.

Two iPAQs were used as Relay 1 and 2, and one of these relay nodes would be elected by OLSR as an MPR. All the traffic generated by Node 1 would then go through that MPR relay node and finally be received by Node 2. Our idea was to terminate the OLSR daemon running on the relay node designated as the MPR. Node 1 would then have to find an alternate route to Node 2 (i.e. through the other relay node). This termination would result in a sudden loss of packets and would be registered on the destination node (Node 2). The convergence time would then be the gap between transmissions, as shown below in Figure 5.4.

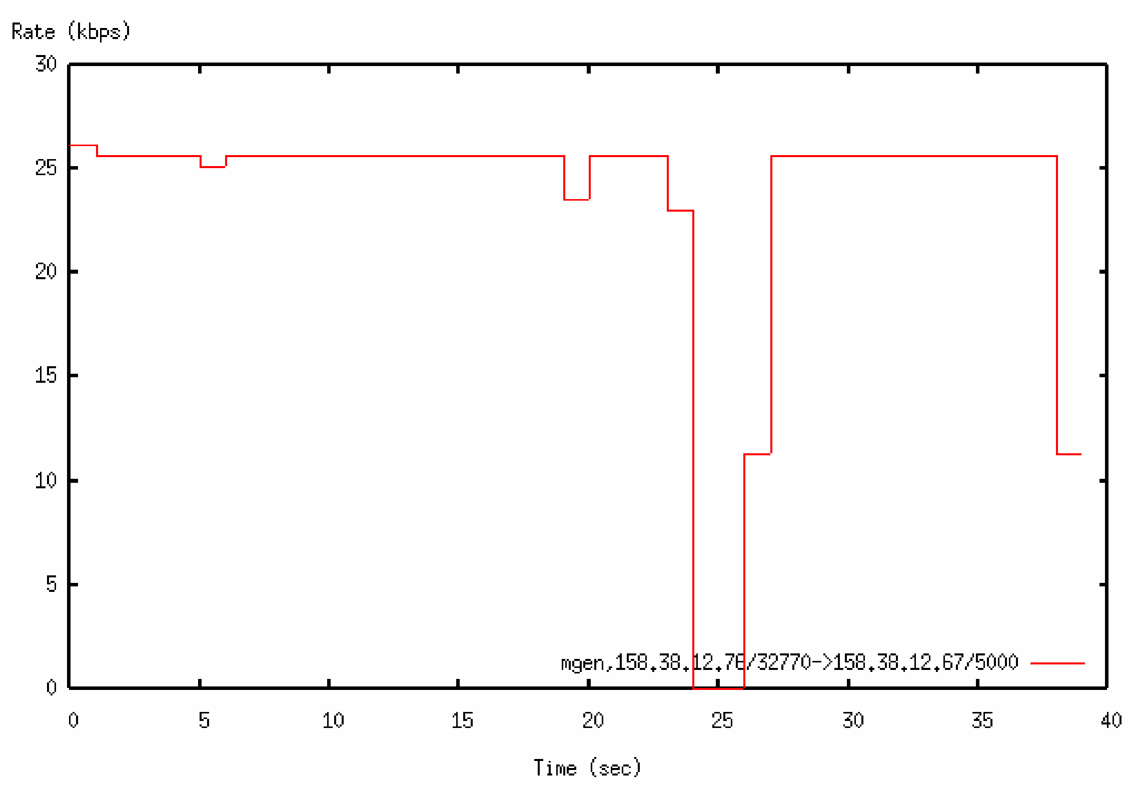


Figure 5.4 – Transmission graph recorded by DREC

This graph shows the result from one of the tests we performed, and the convergence time is represented by the big gap around the 25-second marker.

We used two different protocol configurations; default parameters and modified interval parameters[7]. Each configuration was run three consecutive times, in order to be able to calculate an average result.

[7] Two protocol parameters were changed; `hello_interval` and `tc_interval` (–hint and –tcint). hint=0.25 and tcint=1.0. The details behind this modification are written in Appendix B.2.

All six graphs are located in Appendix B.2 for further study, since they would occupy too much space to be presented here. We will however present the final result, the average convergence time for the two configurations.

Configuration	Average convergence time (s)
OLSR using default parameters	2,8
OLSR using modified parameters[7]	2,7

Table 5.4 – Average convergence time for OLSR
This table shows the final result from our tests. The convergence time is displayed for each configuration.

As this table shows, our results differ very much from those presented in [VGE03] (as shown in Table 5.3). However it is very difficult to explain the variance between the two tests, as [VGE03] have neither specified the setting in which they executed the test, nor how they measured the convergence time. Therefore we contacted Mr. Berner Vegge of the Applica AS Company (responsible for the test presented in [VGE03]).

According to the complete test report from Applica [Veg03], they used a similar test network, but the nodes in their network were positioned in a closer proximity to each other, and they also used five nodes instead of four. They used the RUDE and CRUDE [LSP02] applications to generate and collect data traffic and these tools are similar to MGEN and DREC that we used. However, the most significant difference between the two tests is the fact that the Applica OLSR test used the parameter values as specified in the OLSR manual [LM03], and it would seem that the Naval Research Laboratory (NRL) have changed the default parameters of the OLSR daemon they distribute, without making the appropriate modifications to the manual. Hence our test with default parameters was different from Applica's test. The manual specifies a *Topology Control (TC) timeout multiplier* of 10 seconds, which together with a *TC interval* of 2 seconds gave a *topology hold time* of 20 seconds in the Applica test, while we had a topology hold time of 10 seconds in our default settings. This would explain the big difference in the result from the two tests, as Applica used a more static topology setting.

5.4.3 SELECTING OLSR

The current communication part of the FieldCare application requires only minimal bandwidth, so the increased use of bandwidth of the proactive routing protocols (e.g. OLSR) does not pose as a significant problem in our setting, when video and audio streaming are not considered a part of the application. Anyway, an easy upgrade, in the future, of the network hardware to e.g. the 802.11g standard or other standards will probably provide more than enough bandwidth for FieldCare.

Since the task of incorporating a dynamic routing protocol into the FieldCare application was just a subsection of the entire thesis description, we didn't have enough time to implement one of the protocols from scratch. Therefore we focused on existing implementations of MANET routing protocols and modified FieldCare to use this implementation. The details about this modification are written in section 6.5.

Of the protocols presented in 5.2, only OLSR has been widely developed and implemented. Since the proactive routing scheme didn't constitute any bandwidth problems in our setting, the selection fell on OLSR.

The OLSR implementation we installed on all WLAN devices was developed by the Naval Research Laboratory (NRL)[8] [WDM$^+$03], and further details about this implementation (e.g. installation guide) is given in Appendix A.

5.5 DISCUSSION

In a FieldCare scenario, the most crucial element is the protocol's capability of handling rapid changes in the network topology while at same time provides the necessary mechanisms for securing the routing control data. This is a security requirement we also specified in section 3.4, and as we stated in Table 3.3, the threat of incorrect routing represents a high risk level. Unfortunately, subsection 5.3.2 clearly showed the lack of security in the design of the current dynamic routing protocols. This is usually the case with security; it is realized as a necessity only first when the lack of it has proved disastrous.

However, an interesting option is the SAODV protocol which is both a reactive protocol and has implemented various security measures. A malicious node that plans to build an attack by not behaving in accordance with the routing protocol will only be able to selectively not reply to certain routing messages and to lie about information to itself. Also if a malicious node receives a packet and resends it after a while, it will not cause a topology change because of a sequence number system used by SAODV. Hence, a defense has been created against most of the attacks presented in 5.3.1, with the exception of DoS and jamming attacks. But it is true as the author states in [Zap01], that preventing a DoS attack against the routing protocol in a wireless network is a non-trivial task, as the attacker can simply focus on the physical layer instead. Nevertheless, SAODV is one step in the right direction, concerning network security.

Security implementations for the data transmission are more abundant, though some implementations have serious flaws, like the WEP protocol presented in subsection 5.3.3.1. The 802.11i standard may prove to be the savior of wireless security, but the IETF Working Group "*i*" has not yet completed the work on this standard. In the meantime, transmission security can be added to a higher layer, e.g. by using the SSL (Secure Socket Layer) protocol for end-to-end encryption.

The results of the OLSR test we presented in subsection 5.4.2, shows that this protocol handles topology changes quite well, but the proactive routing scheme used by OLSR is more related to the way traditional routing protocols operate in wired environments. Therefore it is more natural to consider using reactive routing protocols in ad-hoc networks, which handles topology changes very swiftly. Even so, the selection of a routing protocol for FieldCare fell on OLSR since there are more implementations of this protocol than for any of the other presented in section 5.2. And also because the convergence time of OLSR proved to be adequate for a prototype implementation.

[8] NRL's implementation of OLSR is a modification of the INRIA OLSR routing daemon [LM03]. INRIA is the French national institute for research in computer science and control [PL03].

5.6 SUMMARY

The WLAN intended to be used on-site of any disaster area together with the FieldCare application, has to be an ad-hoc network as the area in question may lack the necessary infrastructure. An ad-hoc network is a peer-to-peer network formed by wireless mobile nodes in an arbitrary network topology which is easily deployed in any situation. Given the critical nature of a disaster area, information must flow correctly and securely. A MANET routing protocol handles the communications links and the changes in the network topology. As several proposals exists on routing protocols specifically designed for MANETs, it is important to be aware of the differences between the protocols to be able to select the most suited for the scenario at hand.

We in this chapter reviewed various dynamic routing protocols for mobile ad-hoc networks, and examined the security in such protocols. The Optimized Link State Routing protocol was chosen to be implemented in our new system prototype. Our performance test of this protocol showed that it provided adequate results for a prototype implementation.

However, our future recommendation for the FieldCare project would be to implement the SAODV routing protocol using 802.11i certified devices.

6 IMPLEMENTATION

The implementations we have done in this project are all presented in this chapter. The various sections are arranged in the same order as the description of the specific tasks from 1.1.3. The following list of implementations has been done by us:

- iButton Administration tool
- Access control using iButtons as PiTags
- Dynamic routing
- InfoServer
- Competence Role
- Downloading patient journals to the accident site

6.1 INTRODUCTION

The basis for our implementation work was the FieldCare prototype, developed by SINTEF Telecom and Informatics. A general presentation of the FieldCare system was given in Chapter 2. The sections below provide a description of what we have implemented and also our recommendations for any future work.

The FieldCare prototype was implemented using the Java programming language, so it was natural for us to continue using this language in our implementation[9]. IBM's Eclipse Platform version 2.1 was selected as our software development environment, by recommendation from SINTEF. All the java source code of our implementation is located on the Compact Disc (CD) attached to this report.

We have used the Unified Modeling Language (UML) to describe and present our work, in the form of class diagrams, sequence diagrams and activity diagrams [OMG03].

This chapter will present the implementation we (Group 1) have done, however we will also talk about the system as a whole, since some of the new functionality span over both group's implementations (e.g. acquiring role and downloading patient journals).

6.2 USING LINUX AS AN ALTERNATIVE OS

The FieldCare prototype uses the SavaJe 1.0 operating system[10] on the iPAQs. SINTEF wanted in connection with this project to explore an open source alternative, as SavaJe is a commercial product. We decided to try out Linux as an operating system for all the devices (i.e. MDA, CMDA and FCC), as there exists Linux distributions that support the various system architectures on these devices. The MDAs are HP (Compaq) iPAQs,

[9] It must be noted that the FieldCare prototype was very poorly documented by SINTEF, thus made it quite difficult for us to understand the system in the beginning.

[10] This is a commercial Java based operating system developed by SavaJe Technologies.

and these PDAs use the StrongARM architecture, while the CMDA and FCC are standard PCs with an x86 architecture.

Linux Red Hat 9.0 was installed on both the CMDA and the FCC, and on the MDAs we installed Linux Familiar 0.6. The Linux Familiar project is a part of Handhelds.org, an open source movement sponsored by Hewlett-Packard (HP), which focuses specifically on handheld and wearable computers.

All the technical details regarding the installation of Linux on different devices are located in Appendix A. Figure 6.1 below shows one of the iPAQs we used with Linux installed and the FieldCare application running.

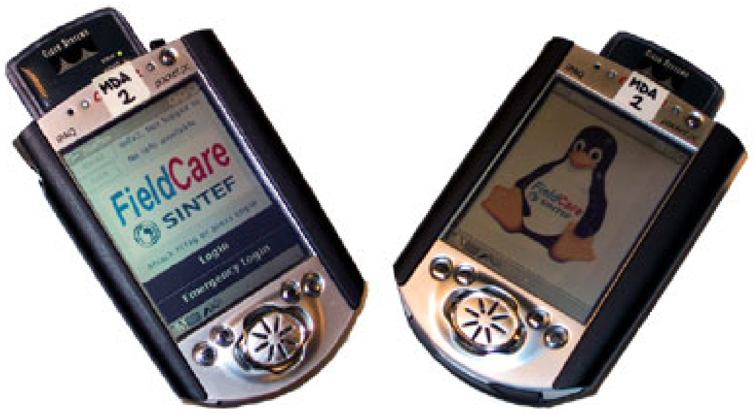


Figure 6.1 – FieldCare on Linux
The iPAQ on the left is running FieldCare on top of a Linux base, and the iPAQ on the right shows the X-Window system with a background image.

6.2.1 SECURITY ISSUES IN OPEN SYSTEM SOFTWARE

There has always been some form of bickering between the two camps in software development; the open source software (OSS) and the commercial off-the-shelf (COTS). Open source software is software with its source code available that may be used, copied, and distributed with or without modifications, and that may be offered either with or without a fee [Ken01], while commercial off-the-shelf software has a closed and proprietary source code distributed in a binary format and sold through commercial licenses.

Open source proponents say open code is inherently more secure because developers can review, test, and fortify it. They believe proprietary software is weaker because a (theoretically) smaller pool of expertise developed it. In contrast, proponents of closed, proprietary systems argue that when the source code is open, it is open to everyone, including those who intend to exploit any weaknesses. This side of the debate believes only a small, trusted development team, such as in a software company, can turn out a trustworthy product [SAZ02].

In July 2001, the MITRE Corporation released a report, drafted for the United States Army, which studies and analyzes the applicability of Linux in a military business case [Ken01]. They presented the following security benefits of OSS.

- An opportunity for interested parties to apply static analysis tools to detect the presence of malicious code or undocumented features. Automated tools can be applied to reduce the effort involved in looking for vulnerabilities.
- Flaws and bugs found in the software can be quickly removed through the creation and distribution of software patches.
- Since the source code is widely available, "white box" testing methods can be performed by interested parties other then just by the developer.
- The wider availability of source code makes it more likely that negative or unexpected consequences of component modification on the rest of the system will be uncovered. This is due to the size and diversity of interested parties able to assess the impacts.
- Individual user organizations can modify the source code to meet their own specific needs.

They also points out that OSS often lacks key security features that are needed to protect critical information and processing. However, they conclude that a poorly configured and managed system is generally insecure, and this is independent of whether or not the system is open source.

For security audits, consultants prefer Linux over Sun Solaris by a ratio of 50:1, according to Network Associates [Cla99]. And a working group comprised of the Defense Advanced Research Projects Agency (DARPA), the General Services Administration (GSA), NIST and the National Security Agency (NSA) concluded that the use of OSS can have both positive and negative effects on the security of a system [DGN^{+}99, Ken99].

In June 2002, the Alexis de Tocqueville Institution (ADTI) released a report entitled "Opening the Open Source Debate" [ADTI02], which, in contrary to the MITRE report [Ken99], condemned open source software. However, this report lost a lot of credibility when it was discovered that ADTI was funded by the Microsoft Corporation [Del02], and in addition to this, the report presented very week arguments and it seemed poorly researched, as [Sko02] states.

6.2.2 FUTURE RECOMMENDATION

Our experience with running Linux on the devices, in particular the iPAQs, has been rather successful because of Linux's versatile nature, in contrast to the static environment of the SavaJe system. Using Linux as the operating system made it also very easy to implement an ad-hoc routing protocol to the system. However, the execution of Java applications on Linux is not quite as fast as compared to the Java tailored SavaJe operating system.

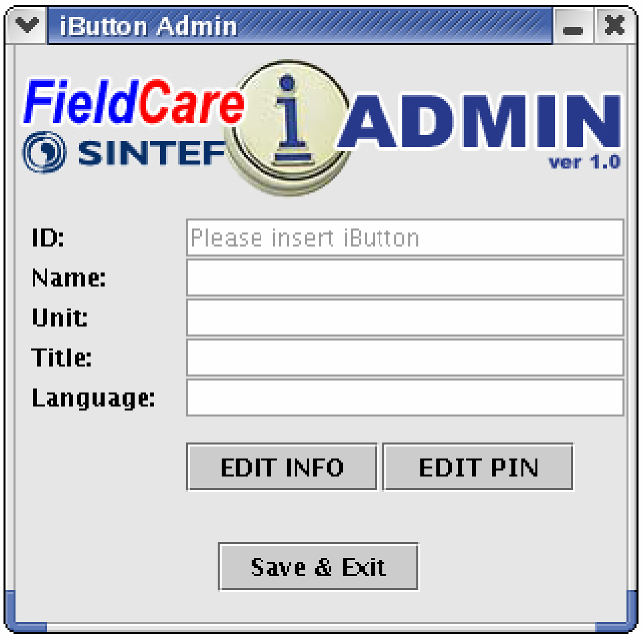
When it comes to the security of Linux, we would go as far as to say Linux is probably the best choice for securing the devices. The reason is that there exists such a great variety of options for securing Linux (e.g. it wouldn't be any problem installing an advanced firewall on an iPAQ running Linux. Try doing that in SavaJe!), however it is far from the easiest choice, as Linux is not a point-and-click experience. And if we may go back to the discussion in 6.2.1 regarding the security in open source system, it is a well known fact that "security by obscurity" is, among many security experts, generally regarded as a non-optimal solution, hence the recommendation for using open systems.

6.3 IBUTTON ADMINISTRATION

In FieldCare, iButtons are used as personal information tags (PiTag). We take these tags in use in the new access control. More information on access control in the FieldCare application is found in section 6.4. The iButton Admin program (IBA) was designed and implemented by us and is used when issuing new PiTags. In this process, user information and a personal PIN code for authentication is stored in the iButton's memory.

6.3.1 OVERVIEW

When starting the iButton Admin, the user will be presented with a graphical user interface (GUI) that tells the user to insert an iButton. The program has at this time made communication with iButtons through the 1-Wire-interface possible, and it is listening for new devices on this interface. When an iButton is inserted, the listener catches this event and sends information about it to the program.



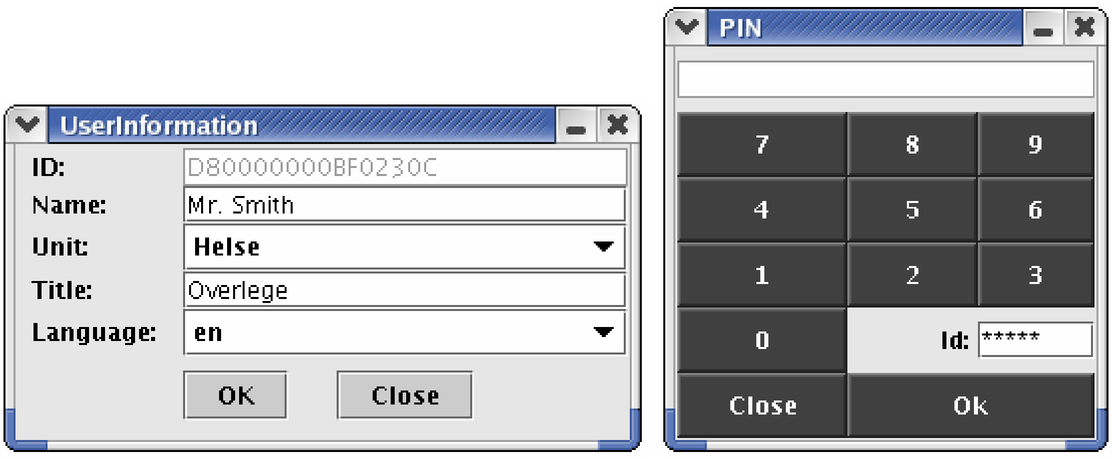
ID:	Please insert iButton
Name:	
Unit:	
Title:	
Language:	

Figure 6.2 – The iButton Admin GUI
This figure shows the Admin GUI. If information is retrieved from the iButton, it will be displayed here.

The program will at this time go through some tests to see if any information is stored on it. If the inserted iButton contains no information, the user will be presented with the *User Information* GUI and the *PIN* GUI coherently. When all the information has been entered, the user will be taken back to the Admin GUI, where he/she may edit the information if any mistakes were made. If information was initially found on the iButton, it will be displayed in the Admin GUI (see Figure 6.2). The ID is retrieved from the iButton itself and is not stored in the memory. Since the iButton now contains some information, the program will let the user edit this information by pressing one of the edit-buttons. The user might want to change the preferred language, the title or maybe he/she has forgotten the PIN code.

If the "edit info" button is pressed, a GUI with user information will be displayed with the existing information. The user may then edit this information and correct any errors, see Figure 6.3. The information in the unit dropdown box is collected from HSDB. The unit information is stored in a database to ease the management of this data, without having to recompile the whole program. When the user is done editing and has pressed "ok", the Admin GUI will reappear.

If the user presses the *EDIT PIN* button, a PIN GUI will be displayed, see Figure 6.3. Here the user has the opportunity to change the PIN. This PIN has to be five digits in length, and the Admin program will not accept any other length. The PIN has to be entered twice for verification. In case of a mismatch, the program will let the user try again. When the PIN is verified, the user will be brought back to the Admin GUI.



UserInformation

ID: D80000000BF0230C
Name: Mr. Smith
Unit: Helse
Title: Overlege
Language: en

PIN

Id: *****

Figure 6.3 – The User Information and PIN GUI
These GUIs are displayed when the user wants to edit information or there is no information stored.

The user may still want to correct come erroneous data, otherwise the user should press *Save & Exit*. The program then takes all the information and stores it on the iButton. The PIN is never stored on the iButton, but rather is hashed with SHA-1 (see section 4.6.2 for more information about this algorithm) and then this hash value is stored instead. The program then connects to the database on the hospital server (HSDB) and stores the information there as well. The information is stored in this database so that we may access it later, when acquiring a role. As default the user is associated with an (lowest privileges) access level 1. This access level may be changed using the RMS (see 2.4) application developed by Group 2.

6.3.2 DETAILED INFORMATION

This section provides more detailed information on how the iButton Admin works through the use of UML diagrams.

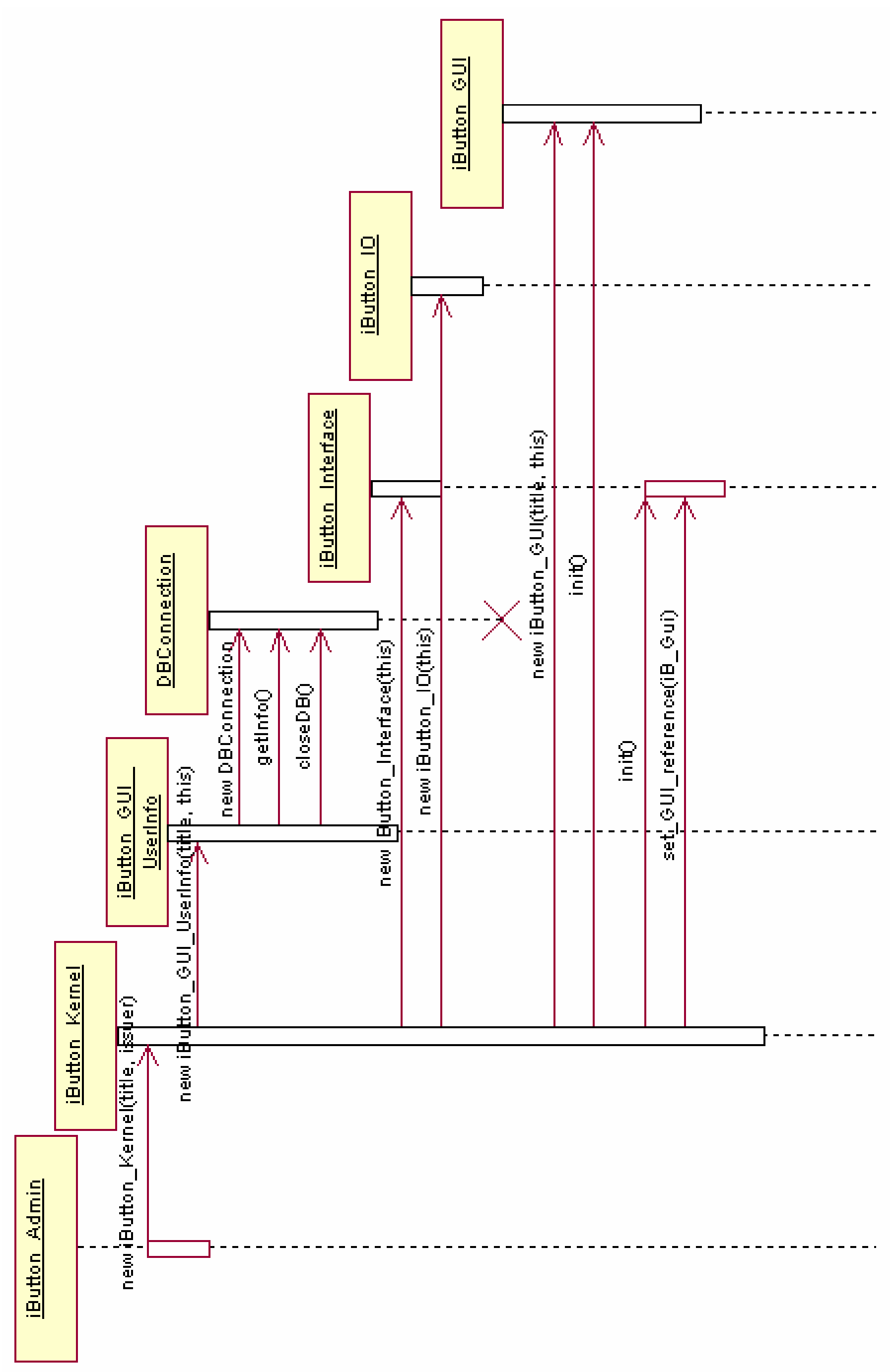


Figure 6.4 – iButton Admin sequence diagram
This figure shows the startup sequence of the iButton Admin.

When the iButton Admin program starts it will instantiate several objects. This is shown in Figure 6.4 above. When this startup sequence is finished, the program displays the Admin GUI (see Figure 6.2) and starts listening for input from the user.

The expected action when the Admin GUI is displayed is that an iButton is inserted or that the user presses “save & exit”. Suppose the user press the *Save & Exit* button, the program will do a check of the information to see whether it is complete or not. In the case where no iButton has been inserted, the information is incomplete, and the program will exit. This should be the case at this time. An overview of the program flow and execution is shown in the activity diagram, Figure 6.5.

If an iButton is inserted, the program will first check whether a file system exists or not on the iButton. Then it will check if the information file, called `STAT.005`, exists. If there is no file system, the iButton will automatically be formatted. The program will consider the iButton as a new device if there is no information file on the iButton, and in that case the program will display the User Information GUI, explained in the previous section, with empty input fields, so that the user may enter the information. In the case where the information file already exists, the program will read the data in this file and update the GUI accordingly. The edit buttons will also now be enabled, and the program will then wait for user interaction (which explained later in this section).

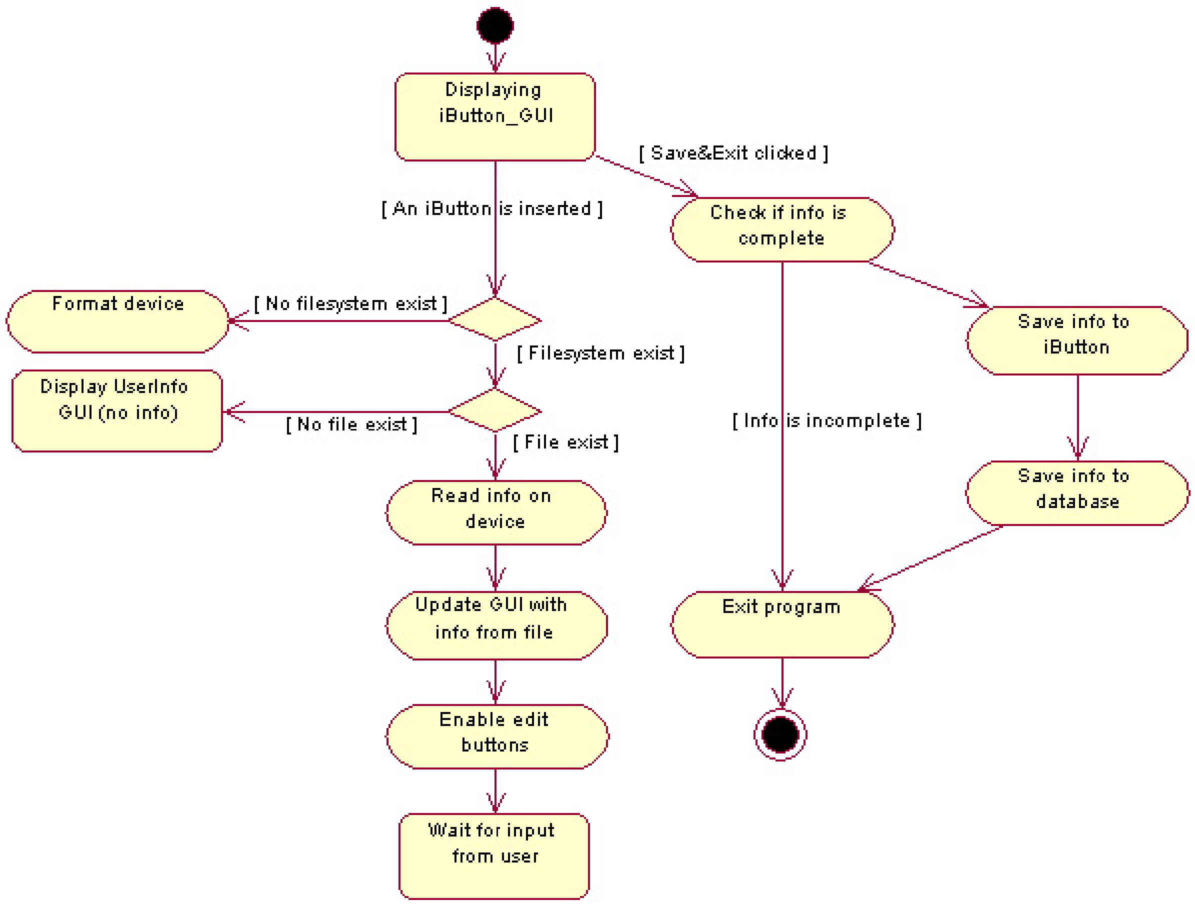


Figure 6.5 – iButton Admin UML activity diagram 1
This figure gives an overview of the program flow after the program has started.

When the program is unable to find an information file on the iButton, it will display the User Information GUI. This GUI lets the user enter his/her data. Since there initially was no information stored on the iButton it is considered as a new device, like we mentioned above. The program flow from this point is shown in Figure 6.6.

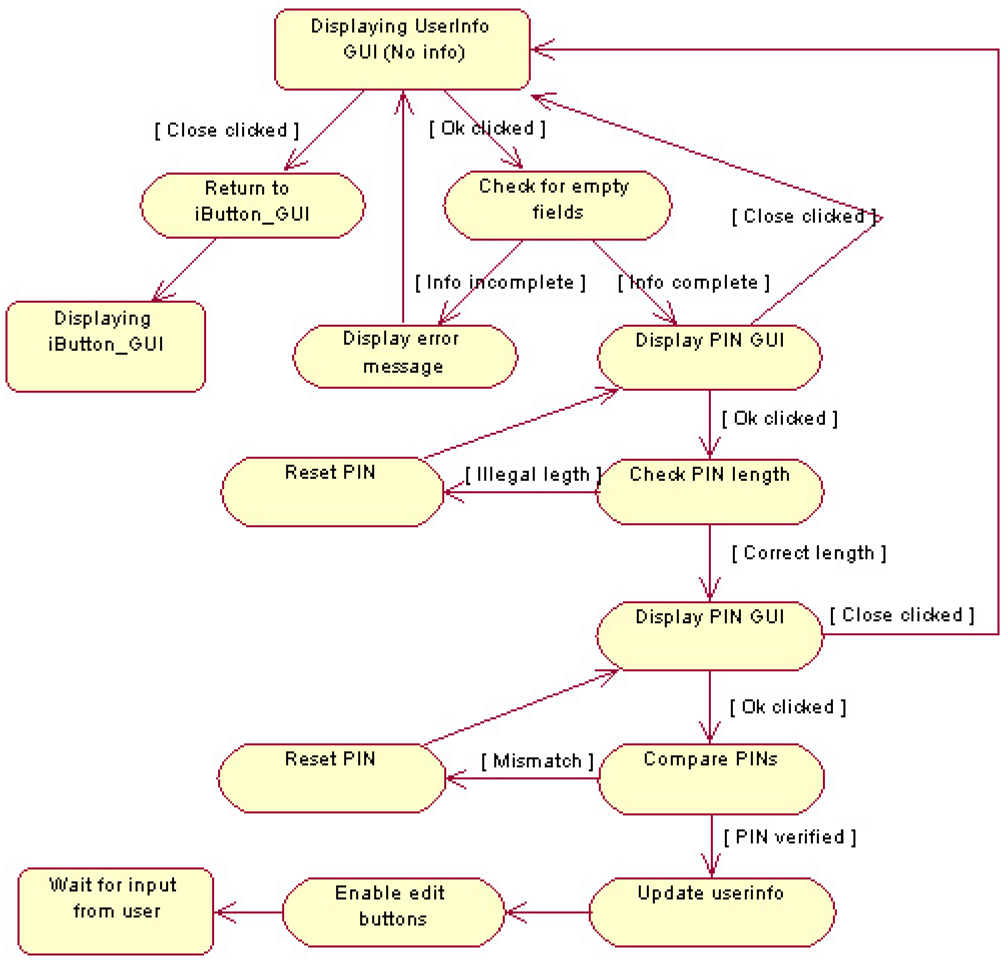


Figure 6.6 – iButton Admin UML activity diagram 2
This diagram show the program flow when the program doesn't detect an information file.

When the User Information GUI is displayed the user has the opportunity to close the GUI. This will bring the user back to the Admin GUI, where an iButton may be inserted or *Save & Exit* may be clicked.

The expected action from the user is that he or she now enters the information. After this the program will check if any of the fields have been left empty, and in such a case display an error message. In the case that the information is complete, the program will display a PIN GUI. Here, the user also has the opportunity to close, and again this will bring him/her back to the Admin GUI. However, if the user acts as expected and enters a PIN, the program will check if this PIN is of correct length; five digits. If the PIN has an invalid length, the user will be notified by a message and has to enter the PIN again. When the PIN is of a correct length, the program caches the PIN, resets the GUI and asks the user to verify the PIN by enter it again. Should there be a mismatch, the user may try again. When the PIN is verified, the program updates the user information, enables the edit buttons in the Admin GUI and brings the user back to this GUI and awaits user interaction.

If the iButton contained information or the user has entered new information, the program now waits for input. The program flow in Figure 6.7 shows how the program reacts to different inputs from the user.

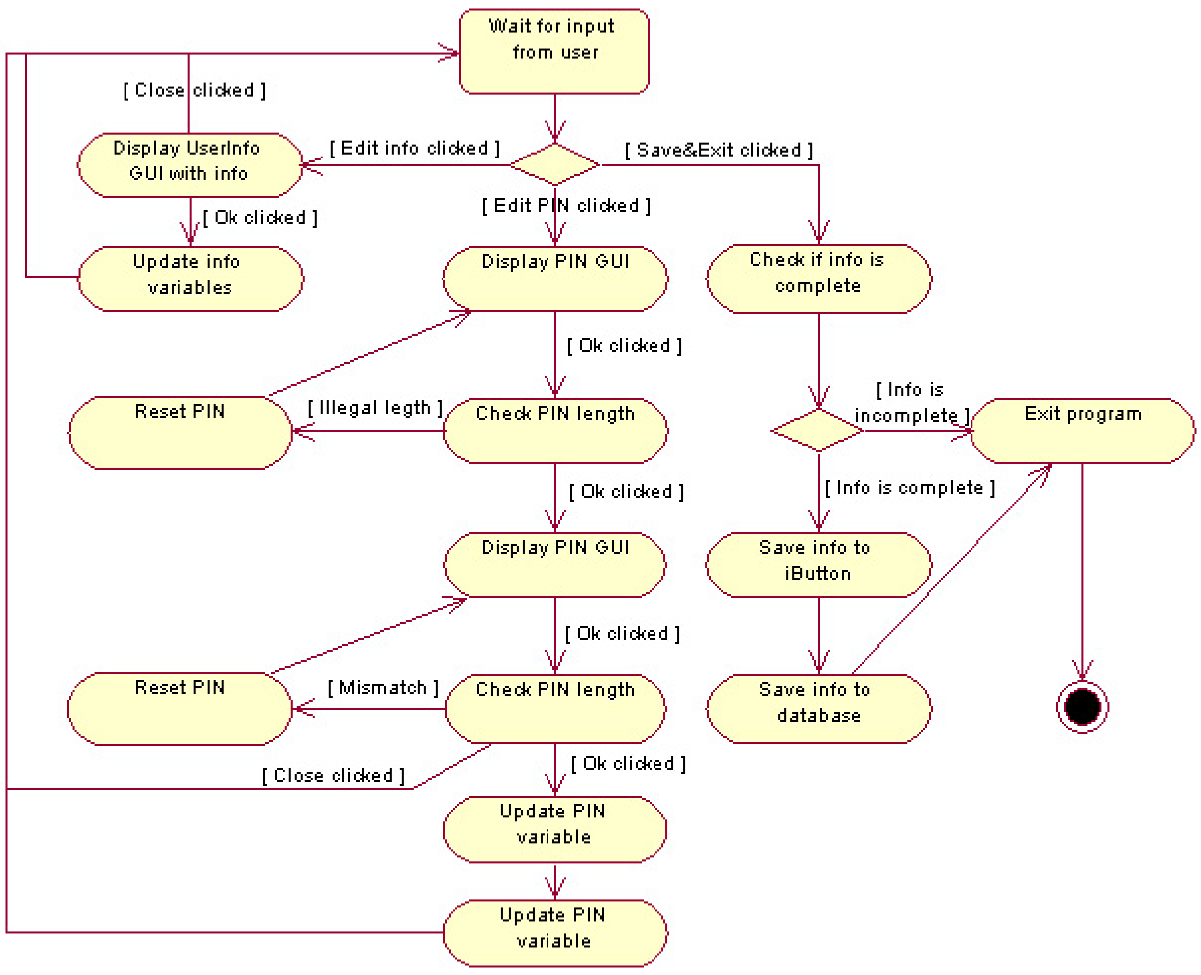


Figure 6.7 – iButton Admin UML activity diagram 3
This figure shows the program flow from the point where the iButton contains information or the user has entered new information.

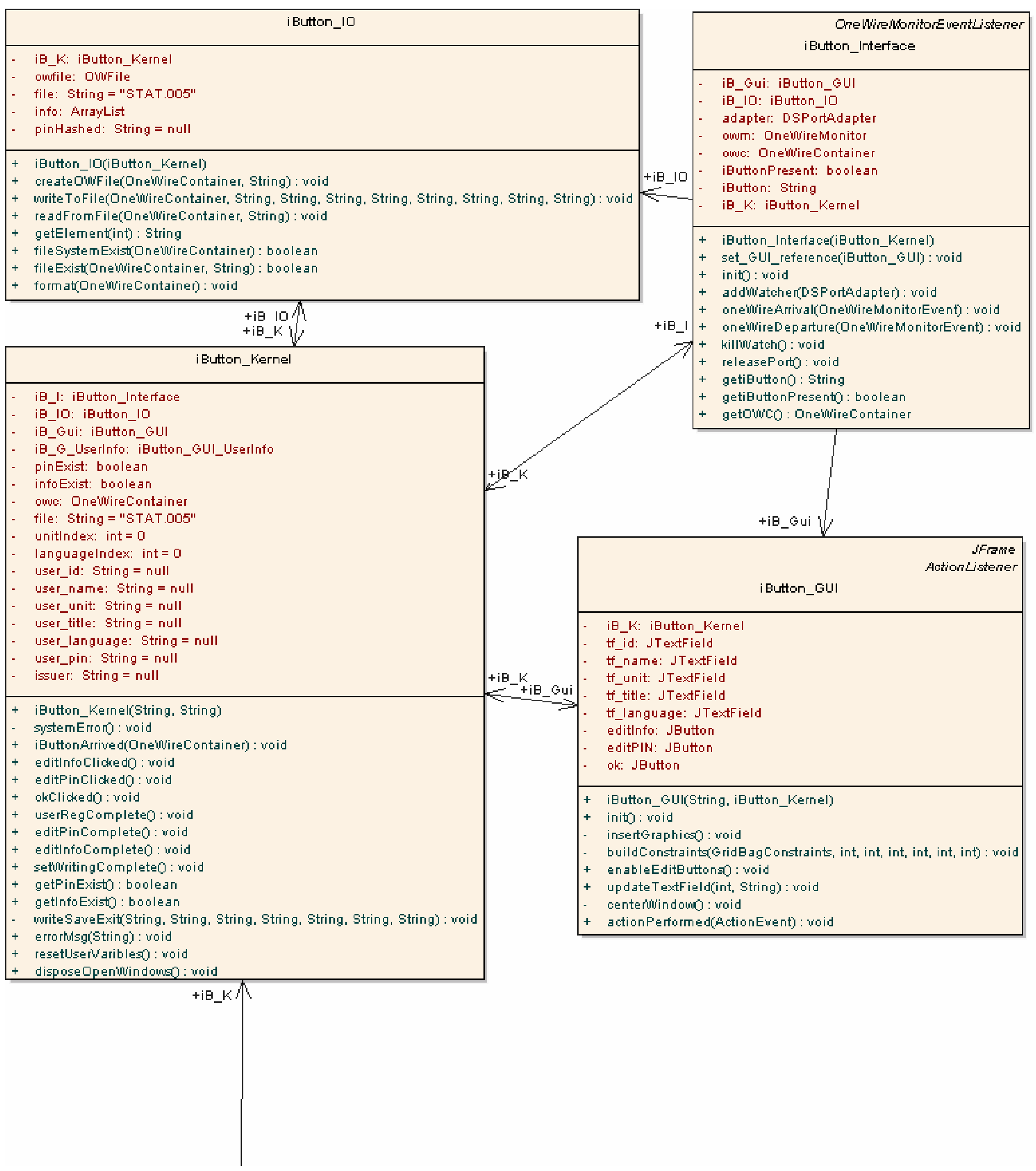
If the user presses the *EDIT INFO* button, the User Information GUI will be displayed. This GUI will contain all the information collected from the iButton or from a user in an earlier stage. As previously explained the user may now enter new or modify the information.

Should the user press the *EDIT PIN* button, the PIN GUI will be displayed, and the user will go through the PIN registration process, as described above.

When the user presses the *Save & Exit* button, the program will perform the checks that was presented in Figure 6.6, and use a hash function to make a message digest of the PIN before all the information is stored on the HSDB and finally the same data are written to the iButton's memory.

6.3.3 IBUTTON ADMIN UML CLASS DIAGRAM

Figure 6.8 below shows an UML class diagram of the iButton Admin program.


iButton_IO
- iB_K: iButton_Kernel
- owfile: OWFile
- file: String = "STAT.005"
- info: ArrayList
- pinHashed: String = null
+ iButton_IO(iButton_Kernel)
+ createOWFile(OneWireContainer, String) : void
+ writeToFile(OneWireContainer, String, String, String, String, String, String, String, String) : void
+ readFromFile(OneWireContainer, String) : void
+ getElement(int) : String
+ fileSystemExist(OneWireContainer) : boolean
+ fileExist(OneWireContainer, String) : boolean
+ format(OneWireContainer) : void
OneWireMonitorEventListener
iButton_Interface
- iB_Gui: iButton_GUI
- iB_IO: iButton_IO
- adapter: DSPortAdapter
- owm: OneWireMonitor
- owc: OneWireContainer
- iButtonPresent: boolean
- iButton: String
- iB_K: iButton_Kernel
+ iButton_Interface(iButton_Kernel)
+ set_GUI_reference(iButton_GUI) : void
+ init() : void
+ addWatcher(DSPortAdapter) : void
+ oneWireArrival(OneWireMonitorEvent) : void
+ oneWireDeparture(OneWireMonitorEvent) : void
+ killWatch() : void
+ releasePort() : void
+ getiButton() : String
+ getiButtonPresent() : boolean
+ getOWC() : OneWireContainer
+iB_IO
+iB_IO
+iB_K
+iB_I
iButton_Kernel
- iB_I: iButton_Interface
- iB_IO: iButton_IO
- iB_Gui: iButton_GUI
- iB_G_UserInfo: iButton_GUI_UserInfo
- pinExist: boolean
- infoExist: boolean
- owc: OneWireContainer
- file: String = "STAT.005"
- unitIndex: int = 0
- languageIndex: int = 0
- user_id: String = null
- user_name: String = null
- user_unit: String = null
- user_title: String = null
- user_language: String = null
- user_pin: String = null
- issuer: String = null
+ iButton_Kernel(String, String)
- systemError() : void
+ iButtonArrived(OneWireContainer) : void
+ editInfoClicked() : void
+ editPinClicked() : void
+ okClicked() : void
+ userRegComplete() : void
+ editPinComplete() : void
+ editInfoComplete() : void
+ setWritingComplete() : void
+ getPinExist() : boolean
+ getInfoExist() : boolean
- writeSaveExit(String, String, String, String, String, String, String) : void
+ errorMsg(String) : void
+ resetUserVaribles() : void
+ disposeOpenWindows() : void
+iB_K
+iB_Gui
JFrame
ActionListener
iButton_GUI
- iB_K: iButton_Kernel
- tf_id: JTextField
- tf_name: JTextField
- tf_unit: JTextField
- tf_title: JTextField
- tf_language: JTextField
- editInfo: JButton
- editPIN: JButton
- ok: JButton
+ iButton_GUI(String, iButton_Kernel)
+ init() : void
- insertGraphics() : void
- buildConstraints(GridBagConstraints, int, int, int, int, int, int) : void
+ enableEditButtons() : void
+ updateTextField(int, String) : void
- centerWindow() : void
+ actionPerformed(ActionEvent) : void
+iB_K
+iB_Gui
+iB_K

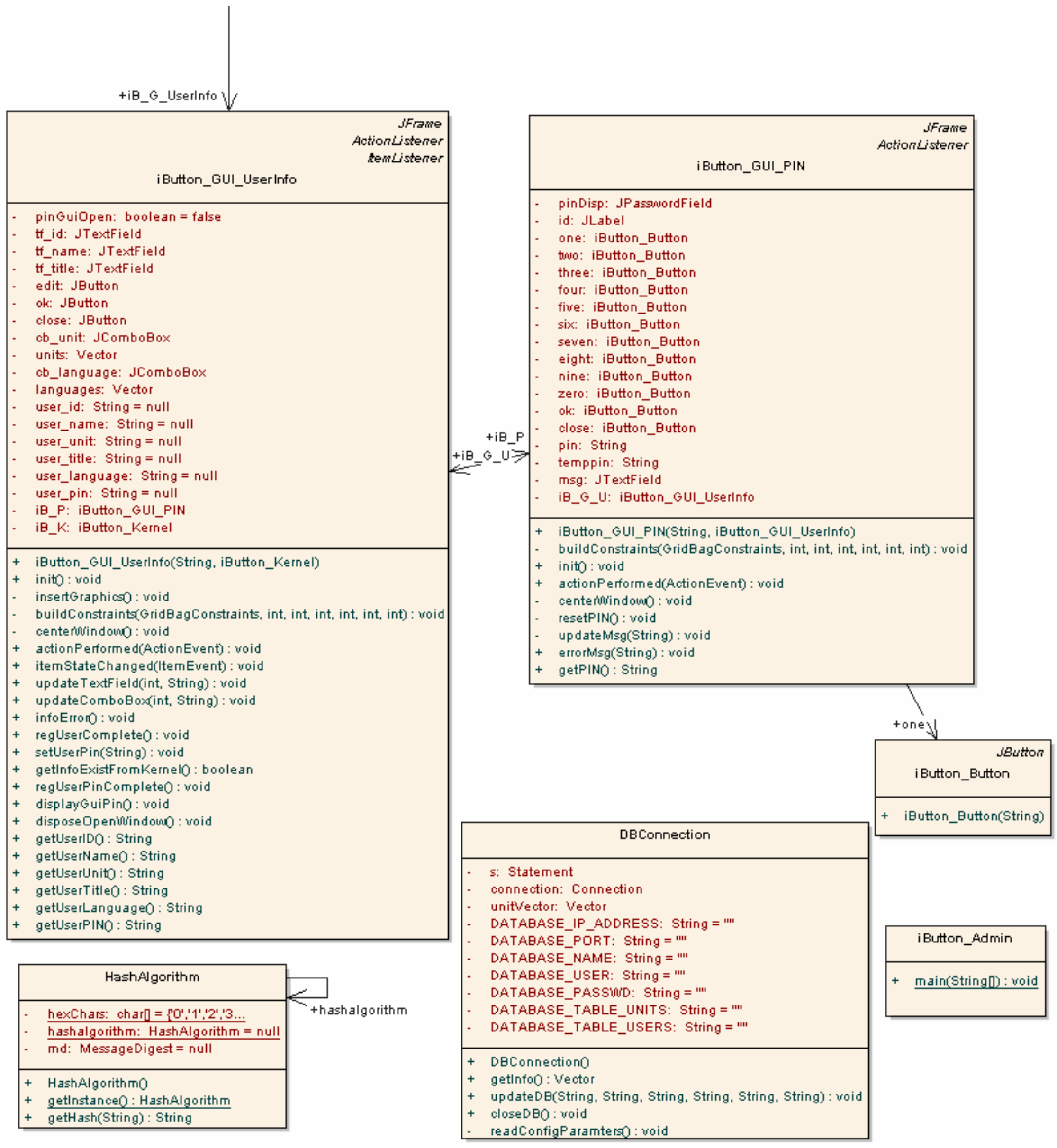

+iB_G_UserInfo
JFrame
ActionListener
ItemListener
iButton_GUI_UserInfo
- pinGuiOpen: boolean = false
- tf_id: JTextField
- tf_name: JTextField
- tf_title: JTextField
- edit: JButton
- ok: JButton
- close: JButton
- cb_unit: JComboBox
- units: Vector
- cb_language: JComboBox
- languages: Vector
- user_id: String = null
- user_name: String = null
- user_unit: String = null
- user_title: String = null
- user_language: String = null
- user_pin: String = null
- iB_P: iButton_GUI_PIN
- iB_K: iButton_Kernel
+ iButton_GUI_UserInfo(String, iButton_Kernel)
+ init() : void
- insertGraphics() : void
- buildConstraints(GridBagConstraints, int, int, int, int, int, int) : void
- centerWindow() : void
+ actionPerformed(ActionEvent) : void
+ itemStateChanged(ItemEvent) : void
+ updateTextField(int, String) : void
+ updateComboBox(int, String) : void
+ infoError() : void
+ regUserComplete() : void
+ setUserPin(String) : void
+ getInfoExistFromKernel() : boolean
+ regUserPinComplete() : void
+ displayGuiPin() : void
+ disposeOpenWindow() : void
+ getUserID() : String
+ getUserName() : String
+ getUserUnit() : String
+ getUserTitle() : String
+ getUserLanguage() : String
+ getUserPIN() : String
+iB_P
+iB_G_U
JFrame
ActionListener
iButton_GUI_PIN
- pinDisp: JPasswordField
- id: JLabel
- one: iButton_Button
- two: iButton_Button
- three: iButton_Button
- four: iButton_Button
- five: iButton_Button
- six: iButton_Button
- seven: iButton_Button
- eight: iButton_Button
- nine: iButton_Button
- zero: iButton_Button
- ok: iButton_Button
- close: iButton_Button
- pin: String
- temppin: String
- msg: JTextField
- iB_G_U: iButton_GUI_UserInfo
+ iButton_GUI_PIN(String, iButton_GUI_UserInfo)
- buildConstraints(GridBagConstraints, int, int, int, int, int, int) : void
+ init() : void
+ actionPerformed(ActionEvent) : void
- centerWindow() : void
- resetPIN() : void
- updateMsg(String) : void
+ errorMsg(String) : void
+ getPIN() : String
+one
JButton
iButton_Button
+ iButton_Button(String)
DBConnection
- s: Statement
- connection: Connection
- unitVector: Vector
- DATABASE_IP_ADDRESS: String = ""
- DATABASE_PORT: String = ""
- DATABASE_NAME: String = ""
- DATABASE_USER: String = ""
- DATABASE_PASSWD: String = ""
- DATABASE_TABLE_UNITS: String = ""
- DATABASE_TABLE_USERS: String = ""
+ DBConnection()
+ getInfo() : Vector
+ updateDB(String, String, String, String, String, String) : void
+ closeDB() : void
- readConfigParamters() : void
iButton_Admin
+ main(String[]) : void
HashAlgorithm
- hexChars: char[] = {'0','1','2','3...
- hashalgorithm: HashAlgorithm = null
- md: MessageDigest = null
+ HashAlgorithm()
+ getInstance() : HashAlgorithm
+ getHash(String) : String
+hashalgorithm

Figure 6.8 – UML class diagram of iButton Admin
This diagram shows all the classes with variables and methods in the iButton Admin program.

6.3.4 FUTURE RECOMMENDATION

There is a problem in the iButton Admin to which we have not yet solved. The program is not capable of detecting a new iButton after another has been written to. This is probably just a small misunderstanding of the 1-Wire API, but we haven't gotten any positive response from the 1-Wire Developer discussion group [Tho03] regarding this problem. Due to this error, we shut down the program after the information is saved, hence the name *Save & Exit*, and you have to restart it in order to another iButton.

6.4 ACCESS CONTROL WITH IBUTTONS

The prototype of FieldCare in which we have based our project on is implemented with three types of login possibilities; using PiTags, PIN code or Emergency Login. This solution of using PiTags, logged the user in, but no authentication was done. When logging on using a PIN code, the PIN was compared to a static variable within FieldCare. This means that all user PINs had to be hard coded in the application, in order to be able to use this login procedure.

We have developed a new access control, combining the usage of PiTags and PIN codes. Our implementation uses iButtons as PiTag, as mentioned in 6.3, and a PIN code for authentication. An overview of the new access control is presented in the next section and a more thorough description is given in 6.4.2.

6.4.1 OVERVIEW

When starting our new FieldCare prototype, the user will be presented with a slightly modified login screen. The difference is that the user now only has two possibilities to log on. He or she can either use a PiTag or use the *Emergency* button, but this *Emergency* mode should, as it says, only be used in an emergency.

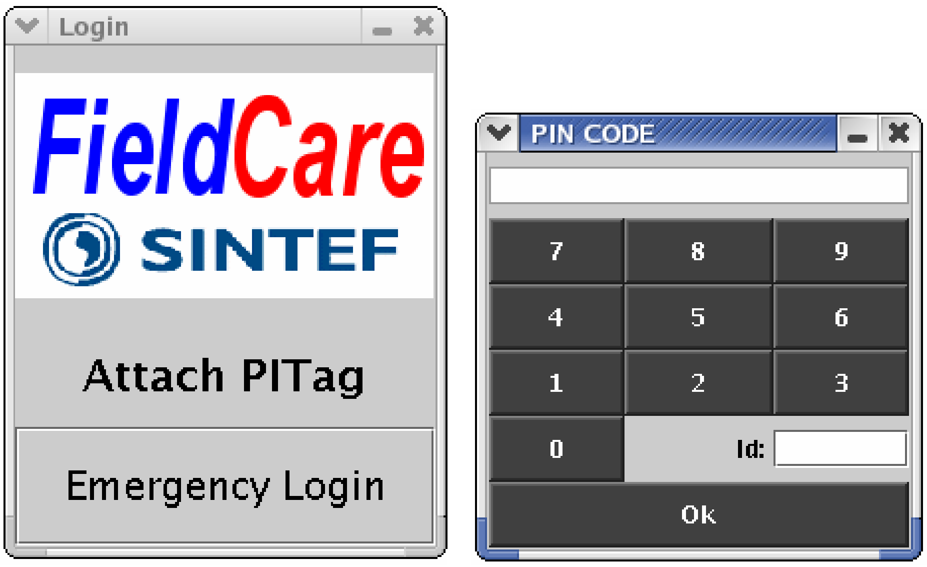


Figure 6.9 – The new login screen and the PIN code panel
This figure shows the new, slightly modified, login screen, and the PIN code panel that is displayed when attaching a PiTag.

When a PiTag is attached, the PIN code panel will be displayed. The PIN, entered by the user, is hashed and compared to the previously hashed PIN stored on the iButton. If the entered PIN should be invalid, the user is allowed trying three times, and if all three tries are invalidated, the user will be logged on in *Emergency* mode. However, if any PIN entered is validated, the FieldCare application will start normally.

6.4.2 DETAILED INFORMATION

As mentioned above, FieldCare will display a new login screen. We have stopped all communication between the devices before a user has been authenticated. In the FieldCare prototype, the communication between devices began immediately after the program had been executed. This was far from an optimal solution, as data traffic started to flow to and from a device managed by a user that had not yet been authenticated. Due to this, we have changed the code in such a way that the device does not start receiving or transmitting data before a user is successfully logged in.

When FieldCare is started, it instantiates the GUI Login which again instantiates GUI Login Interface. This is shown in Figure 6.10 below. The GUI Login Interface handles communication with iButtons through the 1-Wire protocol. When an iButton is attached, the FieldCare will try to read the information file on the iButton. Suppose there is no file, the program will notify the user and exit. This is further discussed in section 6.4.4.

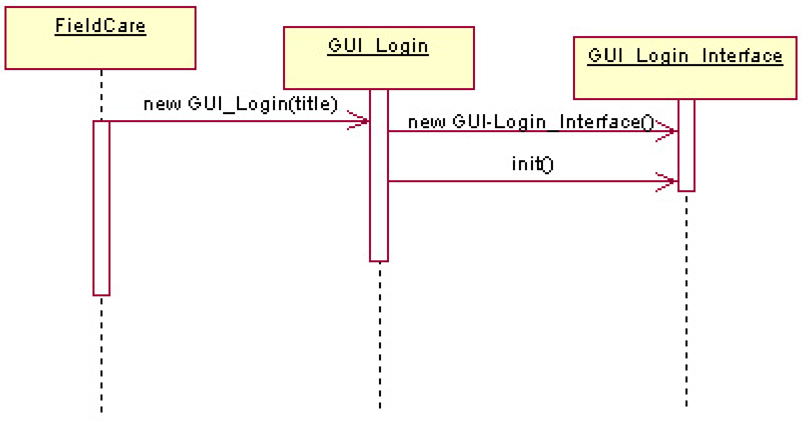


Figure 6.10 – The GUI Login sequence diagram
This figure of the startup sequence shows the classes that are instantiated before the program start waiting for user interaction.

When the user information is found on the iButton, a PIN code panel will be displayed. Here the user has to enter the correct PIN. After the PIN is entered, the program uses SHA-1 to make a hash of the entered PIN, and compares it to the one read from the iButton. If they match, the user is authenticated; otherwise the user has three attempts to enter the correct PIN code. If the PIN is invalidated in the third attempt, the program will log on the user in *Emergency* mode; otherwise *User* mode will be executed. In the Emergency mode the user will not get a Competence Role and are therefore unable to retrieve a patient journal.

The figure below shows an activity diagram of the program when it receives an interaction from the user. After the PIN is verified, the program will start, either in Emergency or User mode. Then the user has to select whether he or she is functioning as a *Medic* or a *Coordinator*.

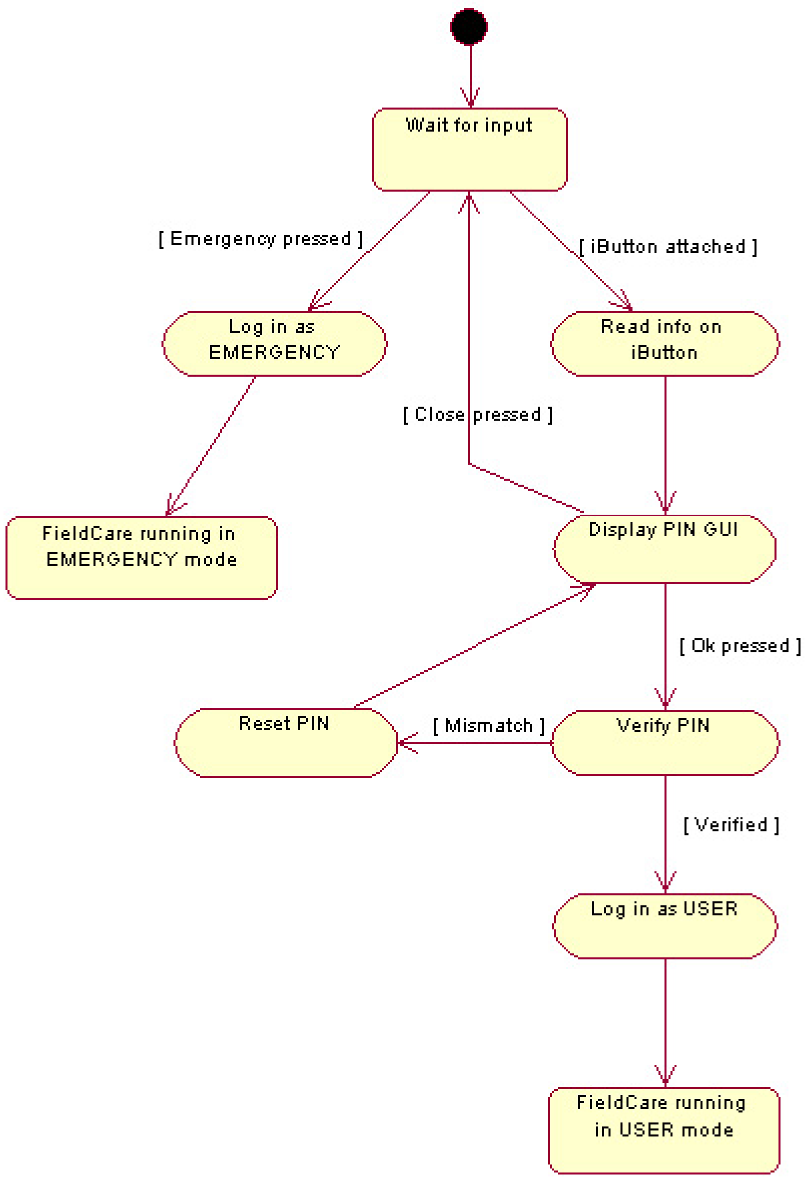


Figure 6.11 – UML activity diagram of FieldCare login
This figure gives an overview of the program flow when the FieldCare application has been started and is waiting for user interaction.

6.4.3 LOGIN CLASS DIAGRAM

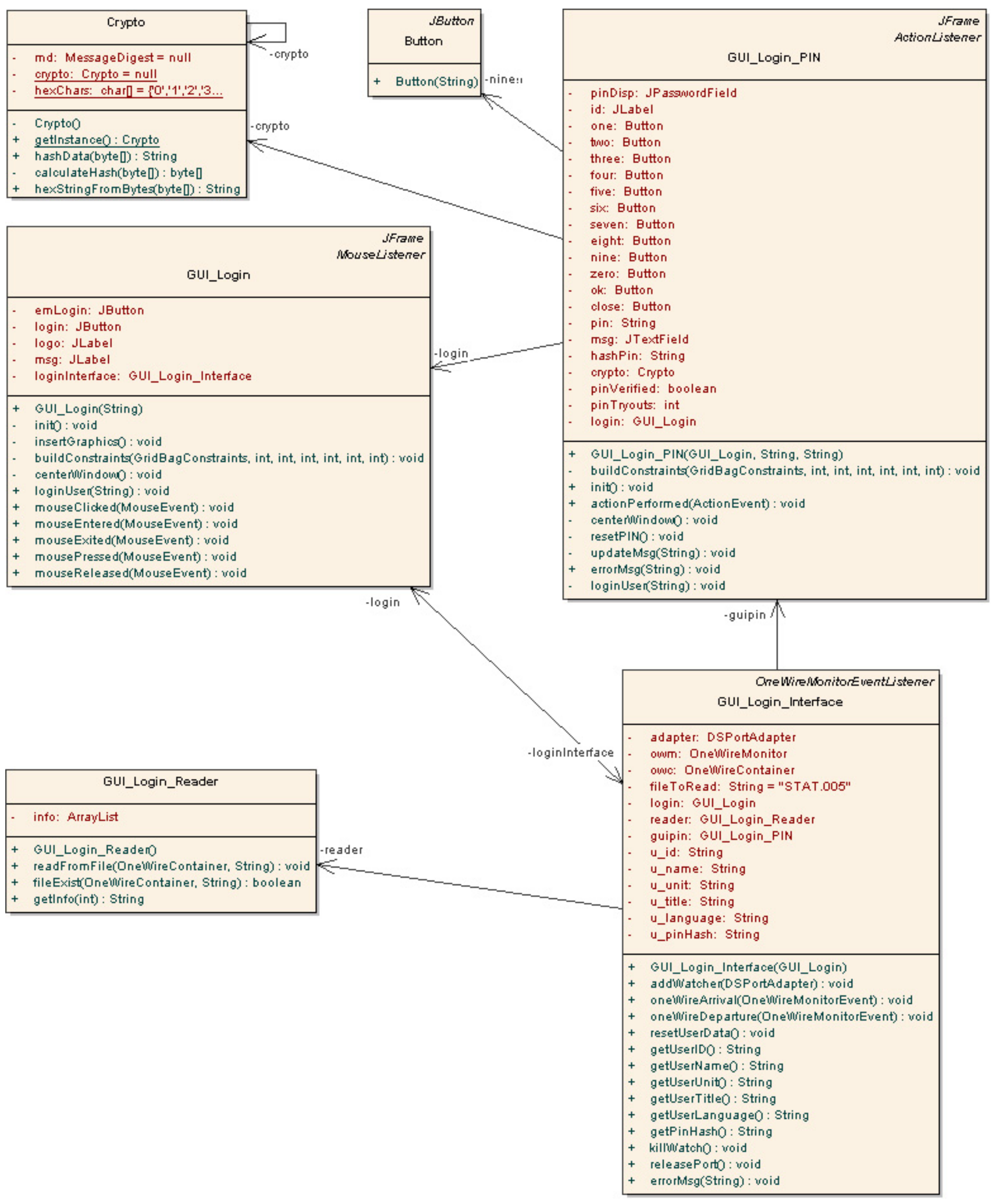

Crypto
- md: MessageDigest = null
- crypto: Crypto = null
- hexChars: char[] = {'0','1','2','3...
- Crypto()
+ getInstance() : Crypto
+ hashData(byte[]) : String
- calculateHash(byte[]) : byte[]
+ hexStringFromBytes(byte[]) : String
-crypto
JButton
Button
+ Button(String)
-nine
JFrame
ActionListener
GUI_Login_PIN
- pinDisp: JPasswordField
- id: JLabel
- one: Button
- two: Button
- three: Button
- four: Button
- five: Button
- six: Button
- seven: Button
- eight: Button
- nine: Button
- zero: Button
- ok: Button
- close: Button
- pin: String
- msg: JTextField
- hashPin: String
- crypto: Crypto
- pinVerified: boolean
- pinTryouts: int
- login: GUI_Login
+ GUI_Login_PIN(GUI_Login, String, String)
- buildConstraints(GridBagConstraints, int, int, int, int, int, int) : void
+ init() : void
+ actionPerformed(ActionEvent) : void
- centerWindow() : void
- resetPIN() : void
- updateMsg(String) : void
+ errorMsg(String) : void
- loginUser(String) : void
JFrame
MouseListener
GUI_Login
- emLogin: JButton
- login: JButton
- logo: JLabel
- msg: JLabel
- loginInterface: GUI_Login_Interface
+ GUI_Login(String)
- init() : void
- insertGraphics() : void
- buildConstraints(GridBagConstraints, int, int, int, int, int, int) : void
- centerWindow() : void
+ loginUser(String) : void
+ mouseClicked(MouseEvent) : void
+ mouseEntered(MouseEvent) : void
+ mouseExited(MouseEvent) : void
+ mousePressed(MouseEvent) : void
+ mouseReleased(MouseEvent) : void
-login
-login
-guipin
-loginInterface
OneWireMonitorEventListener
GUI_Login_Interface
- adapter: DSPortAdapter
- owm: OneWireMonitor
- owc: OneWireContainer
- fileToRead: String = "STAT.005"
- login: GUI_Login
- reader: GUI_Login_Reader
- guipin: GUI_Login_PIN
- u_id: String
- u_name: String
- u_unit: String
- u_title: String
- u_language: String
- u_pinHash: String
+ GUI_Login_Interface(GUI_Login)
+ addWatcher(DSPortAdapter) : void
+ oneWireArrival(OneWireMonitorEvent) : void
+ oneWireDeparture(OneWireMonitorEvent) : void
+ resetUserData() : void
+ getUserID() : String
+ getUserName() : String
+ getUserUnit() : String
+ getUserTitle() : String
+ getUserLanguage() : String
+ getPinHash() : String
+ killWatch() : void
+ releasePort() : void
+ errorMsg(String) : void
GUI_Login_Reader
- info: ArrayList
+ GUI_Login_Reader()
+ readFromFile(OneWireContainer, String) : void
+ fileExist(OneWireContainer, String) : boolean
+ getInfo(int) : String
-reader

Figure 6.12 – UML class diagram of the new login
This diagram shows all the classes with variables and methods in the new login.

6.4.4 FUTURE RECOMMENDATION

As mentioned earlier, the program will exit if an iButton with no information file is attached when the user is logging on. This should be changed so that the user is logged on in "Emergency" mode instead. When logging out, we had to do a small hack. We weren't able to restart the network traffic properly. Due to this we are just shutting down the whole program and starting it again. This is done using a bash script, see Appendix C.4.1. The ultimate solution would be to shut down all network traffic to and from a device when the user logs out. We encountered a problem with the serial port on Linux Familiar, but as this is not related to our implementation, it is presented in A.9.

6.5 ROUTING MANAGEMENT

The FieldCare prototype is implemented with a static routing scheme. All the nodes that you want to participate in the WLAN must be predefined in the `hosts` file, and the topology is set by specifying the peers of each node in the `config` file. We have changed all this, and implemented a dynamic routing scheme by using the ad-hoc routing protocol called OLSR (presented in 5.2.1). This protocol handles rapid topology changes and the discovery of new nodes and routes. After the OLSR daemon[11] has been started, it will automatically update the system's routing table (in our case the Linux routing table). A routing module implemented in FieldCare, which we called *Routing Management*, will then regularly read the routing table, check for changes and update FieldCare accordingly.

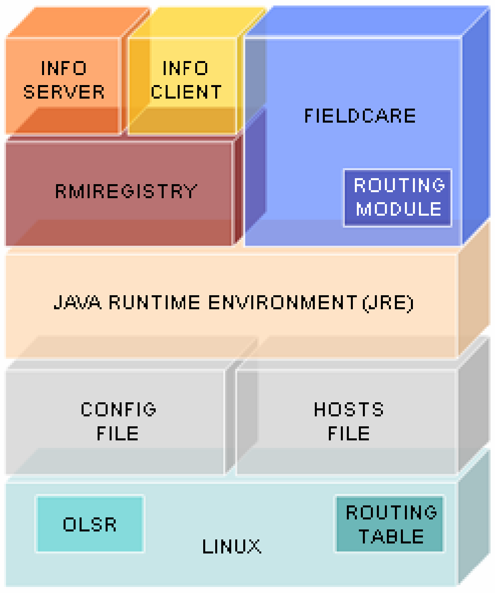


Figure 6.13 – The current application stack
This is how our FieldCare devices are structured in various layers from the operating system in the bottom, to the FieldCare application and the InfoServer system at the top.

Figure 6.13 above, displays how we have structured the FieldCare system on each of the WLAN devices. The bottom layer is new, as we have changed the operating system to Linux. Other new parts include; the OLSR module, the routing table, the routing module within FieldCare and the InfoServer system.

6.5.1 ACTIVITY DIAGRAM

The following UML activity diagram was created to show the operation of the Routing Management class when instantiated.

[11] A piece of software that forks to a background process of the operating system.

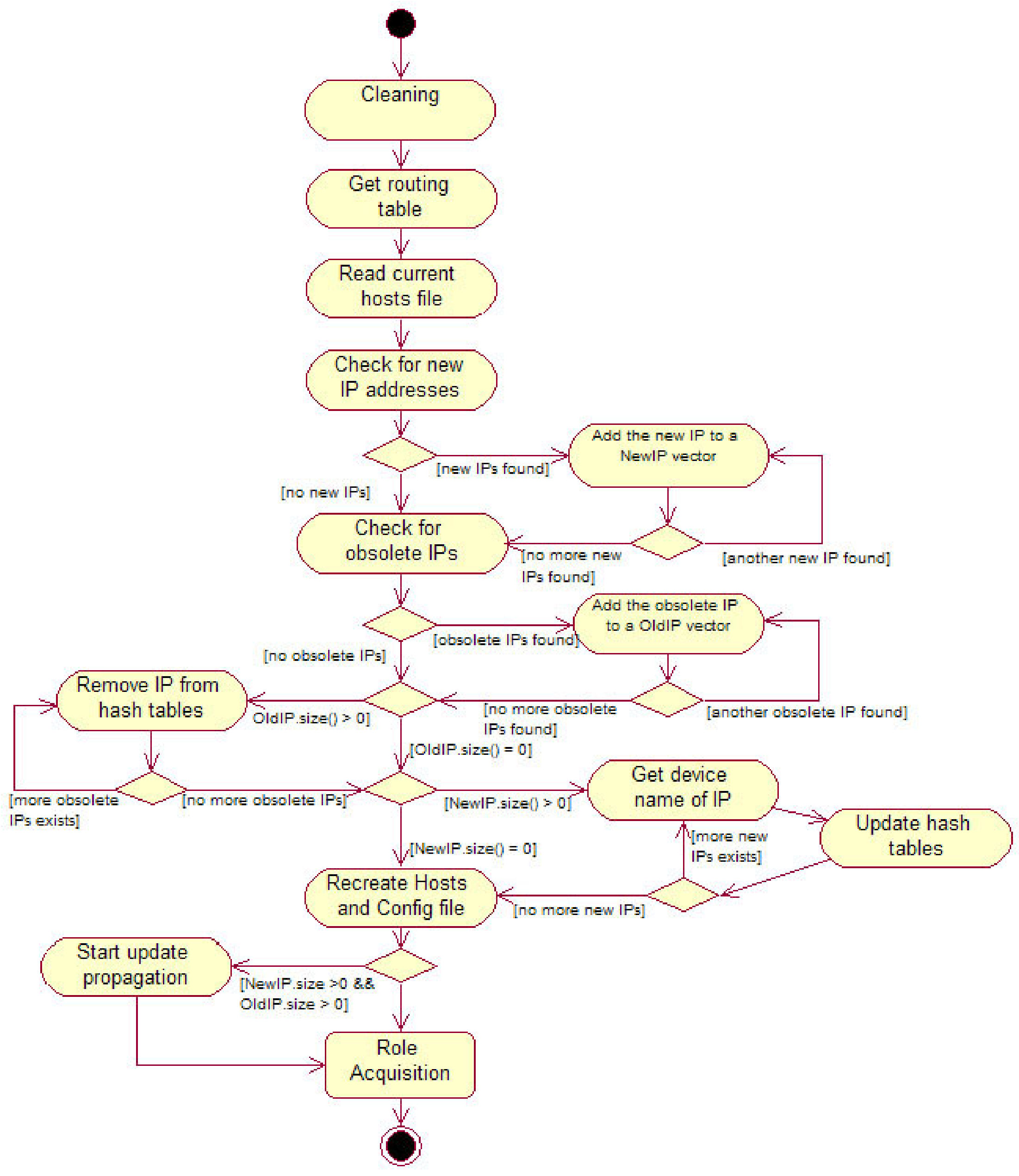


Figure 6.14 – Activity diagram of the Routing Management class
This UML activity diagram shows the flow of logic in the Routing Management class.

The "*Get device name of IP*" action, in Figure 6.14, is described in detail in the following section, while section 6.5.3 describes the "*Start update propagation*" action. The "*Role Acquisition*" state at the end of the diagram is explained in section 6.7.

6.5.2 GETTING REMOTE DEVICE NAMES

The FieldCare prototype we have based our work on, used a static routing scheme where all the nodes were predefined, and the mapping between each node's IP address and its device name was located in the `hosts` file. This is not applicable to our dynamic routing implementation, as the network topology may change continuously.

Since the FieldCare application is, in many cases, dependent on using the name of a device instead of using its IP address, we decided to implement a solution that adapts to this mode of operation. The IP addresses of the devices, currently part of the WLAN, are easily available through the OLSR protocol (which updates the routing table in Linux), but each node's device name is not readily available, as this information is only stored locally on each node. We implemented a client/server system, called *InfoServer* (as presented in section 6.6), which is responsible for processing various remote requests. Each node has installed this InfoServer, and is therefore able to both serve others and make requests of its own.

When the server part of the InfoServer system is started, it will read FieldCare's `config` file, locate the device name specified in this file and import it so that the server is able to answer requests for the device name from other nodes. The Routing Management class in FieldCare will send a device name request to any new nodes that OLSR may discover, in order to get the IP-to-Name (and vice versa) mapping that many of the other classes in FieldCare is dependent on. The sequence diagram in Figure 6.15, displays this process.

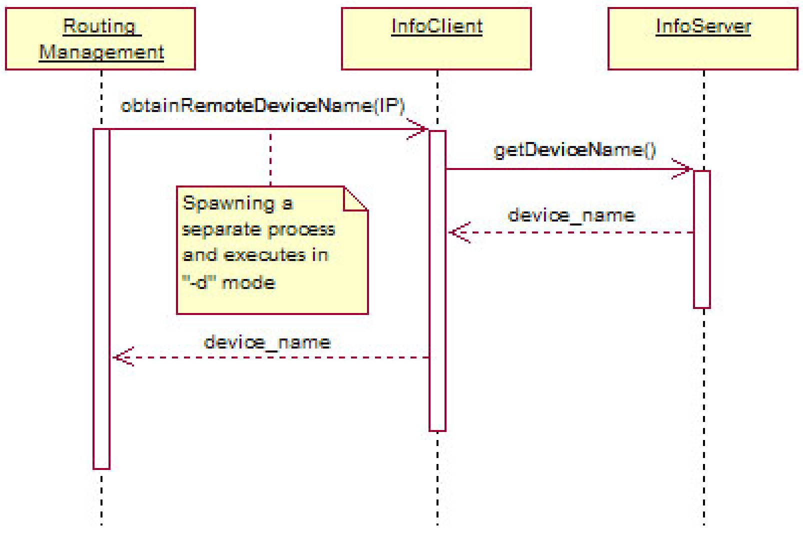

Routing Management
InfoClient
InfoServer
obtainRemoteDeviceName(IP)
getDeviceName()
device_name
Spawning a separate process and executes in "-d" mode
device_name

Figure 6.15 – Get remote device name sequence
This UML sequence diagram shows the message communication between the involved classes when the Routing Management class tries to get the name of a given remote device. Note that the InfoClient is located on the device detecting a new node, while the InfoServer is located on the new node.

We have in retrospect discovered one serious design flaw that could result in an undesired behavior of the FieldCare application. When the InfoClient is unable to get a reply from the remote device's InfoServer, the InfoClient will return the string "unknown", concatenated with a millisecond timestamp to the Routing Management class. During the next run of the Routing Management, if the IP address of the device not responding to the get device name request is still in the routing table, Routing Management will again use the InfoClient to obtain the node's device name. And as this continues, the name and address lists in FieldCare will be filled with invalid "unknown" entries. A simple remedy for this problem is to do a more thorough check of what exactly has been registered in the name and address lists before issuing new requests. However, this problem shouldn't occur if the InfoServer is always started prior to the initialization of the FieldCare application and the OLSR daemon.

6.5.3 PROPAGATION OF UPDATE MESSAGES

When the Route Management class has detected an alteration in the routing table; either a new node or a node removed, then the "*Start update propagation*" action in Figure 6.14 is executed. This sequence of events will update the peers vector, which contains all currently reachable nodes, in those FieldCare classes where this variable is used. The update message propagation is schematically displayed in Figure 6.16.

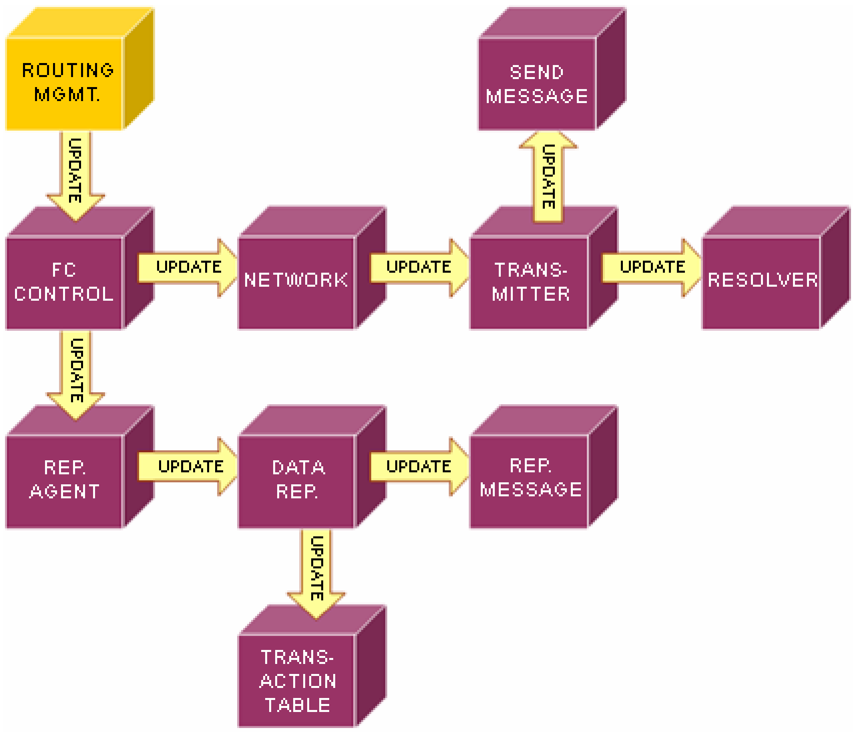


Figure 6.16 – Update propagation
A simple schematic of the update message propagation throughout FieldCare.

Each update message contains a vector that only holds the current address list. This list is actually a hashtable with the IP addresses used as keys for mapping the device names.

The classes *DataReplicator*, *Transmitter* and *TransactionTable* are each responsible for resolving which peers that are new, and should therefore be added, and which peers that are obsolete and hence should be removed. The theory of *Venn* diagrams is applied in this context.

Figure 6.17 below, presents an UML sequence diagram of the update message propagation.

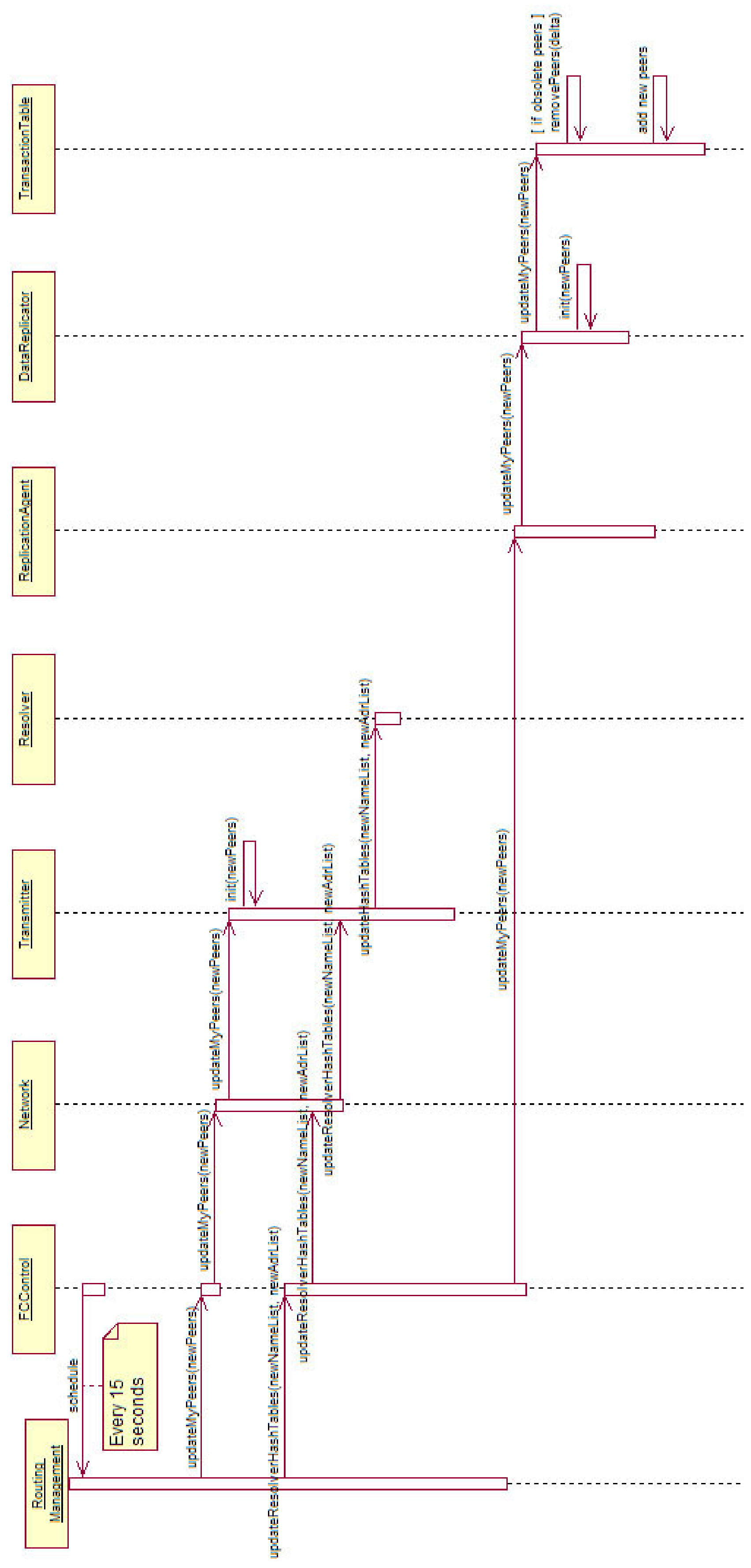


Figure 6.17 – Update propagation sequence diagram

There exists a bug in this implementation, but unfortunately due to lack of time, it has not yet been fixed. When we implemented the dynamic routing, we assumed that the replication strategy in the FieldCare prototype was not dependent on the static network topology used. However, this proved to be an incorrect assumption. The old prototype builds the peers vector during initialization and uses this vector when starting the replication, but this is only done once during startup, and never repeated.

Therefore, in our setting, when the first node is initialized, the replication is started based on an empty peers vector. The second node starting will have one peer; the third will have two peers, and so on. The effect of this is that any node joining the network after some data have been registered and replicated, will not be part of the set used by the other nodes during replication, and hence these new nodes will not receive the existing data. However they will receive all new data being registered.

This problem occurs most likely during the update phase in the ReplicationAgent class and the DataReplicator class, and can be correct by employing a proper management routine, that carefully adds and removes elements from all lists each time these classes are updated.

6.5.4 FUTURE RECOMMENDATION

Apart from fixing the replication bug, we believe it's in SINTEF's interest to investigate if it would be possible to incorporate the OLSR protocol into the FieldCare application. This would probably mean having to implement the protocol in Java. It would also be interesting to implement the recommendation we gave in the summary in section 5.6, regarding routing protocols and security.

6.6 INFOSERVER

The InfoServer system consists of two general elements; a server and a client part. Whenever we refer to the name "*InfoServer*", we mean the whole system with both parts; server and client. However when both InfoServer and InfoClient is used in the same context, we then refer to the server and client components respectively.

All devices on the WLAN, i.e. MDAs, CMDAs and FCC, have installed the InfoServer in addition to the FieldCare application. The InfoServer is responsible for the forwarding and delivery of Competence Roles and patient journals, and also for the delivery of the name of each device. It uses Java's Remote Method Invocation (RMI) in the communication between the nodes in the network.

An UML class diagram of the InfoServer system is shown in Figure 6.20 later on. All sequence diagrams and activity diagrams related to the InfoServer are presented in the sections 6.7 and 6.8. These sections deal with the acquisition scenarios; request and delivery of role and journal.

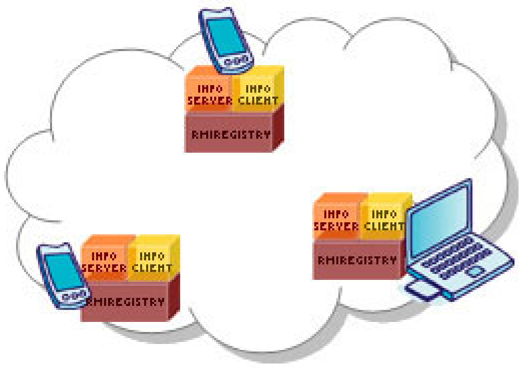


Figure 6.18 – InfoServer on each device in the WLAN
In addition to the FieldCare application installed on each device, the InfoServer is also installed as a separate application. The InfoServer uses the RMI Registry for binding and lookup requests.

6.6.1 EXTERNAL INFOSERVER

Since the FCC device has two network interfaces, and hence two different IP addresses, one for the internal WLAN and another for the link to the FCCDB device (the FCC database machine installed by Group 2), the FCC needs to have two server parts installed, one serving the internal net and the other serving the external net. We have called this second server on the FCC for *External InfoServer*. In essence, the External InfoServer has to perform two tasks; deliver 1) the Competence Role and 2) the patient journal to the InfoServer on the device requesting the information. The services offered by the External InfoServer are only used by the FCCU application, installed by Group 2 on the FCCDB, after the request has been processed by the hospital server database and returned to the FCCDB.

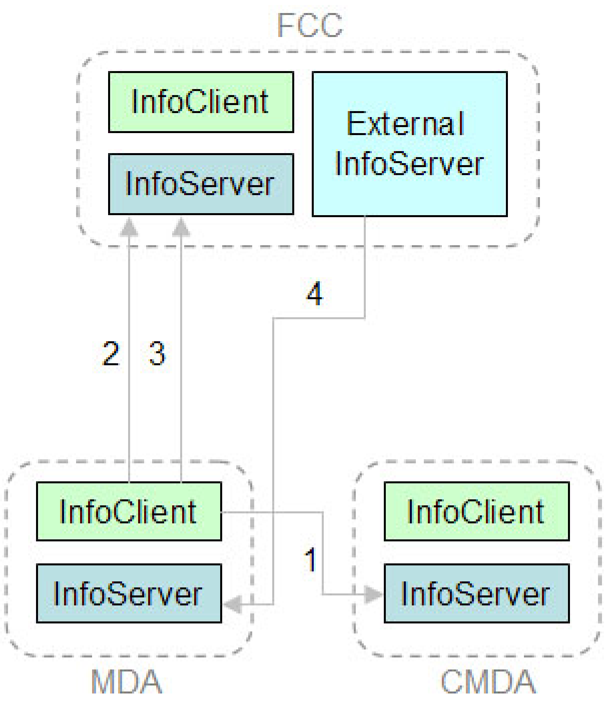


Figure 6.19 – InfoServer intercommunication
This figure shows the various communication sequences intiated when the MDA enters the network.

Figure 6.19 shows the events that are initiated when the MDA enters the network. First (1 and 2) the MDA asks the InfoServer on both CMDA and FCC for their device names, as the OLSR routing protocol only deals with IP addresses. Then (3) the MDA sends a request for a Competence Role to the InfoServer on the FCC, which in turn inserts the request into the database on the FCCDB device. The FCCDB periodically initiates a connection to the hospital server (HSDB) over a GPRS/GSM[12] link, and the request is then processed by the hospital server and a role (if granted) is returned to the FCCU on the FCCDB, which then forwards it to the External InfoServer on the FCC. The External InfoServer will then (4) deliver the role to the InfoServer of the device requesting it (i.e. MDA). This role request sequence is explained in more detail in section 6.7.

6.6.2 SECURE INFOSERVER

Security is paramount, as sensitive data (e.g. a patient's journal) are being transferred over the connections established by the InfoServer. As we discussed in section 3.4, a strong end-to-end encryption must be enforced when dealing with this kind of information. This was also the conclusion of the Data Inspectorate, as we presented in 3.3.

Secure Sockets Layer (SSL) was implemented in the InfoServer by using Sun's Java Secure Socket Extension (JSSE). This security API enables the use of end-to-end (i.e. client to server) symmetric encryption with a 128-bit SSL key and mutual authentication of the server and client.

A 1024-bit RSA key pair was generated and used in the authentication mechanism. The signature algorithm is derived from the algorithm of the underlying private key [Sun01], i.e. in our implementation the signature algorithm was MD5withRSA. We used Sun's Key and Certificate Management Tool, called *keytool*, to generate and prepare the certificates; one for the server and one for the client. The keytool is located in the `%JAVA_HOME/bin` directory.

Generating the keystores:

```
# keytool -genkey -alias infoClient -dname "CN=FieldCare, OU=ITEM,
O=NTNU, L=Trondheim, S=ST, C=NO" -keyalg RSA -keysize 1024 -keypass
eracdleif -storepass eracdleif -keystore .infoClientKeyStore

# keytool -genkey -alias infoServer -dname "CN=FieldCare, OU=ITEM,
O=NTNU, L=Trondheim, S=ST, C=NO" -keyalg RSA -keysize 1024 -keypass
eracdleif -storepass eracdleif -keystore .infoServerKeyStore
```

Exporting the certificates from the keystores to external files:

```
# keytool -export -alias infoClient -storepass eracdleif -file
client.cer -keystore .infoClientKeyStore

# keytool -export -alias infoServer -storepass eracdleif -file
server.cer -keystore .infoServerKeyStore
```

[12] GPRS/GSM was used in the implemented prototype, as this was the only available option. However, alternatives should be considered to provide a higher level of dependability [HMS03].

And finally we import the server's certificate into the client's keystore, and visa versa.

```
# keytool -import -v -trustcacerts -alias Client -file server.cer -
keystore .infoClientKeyStore -keypass eracdleif -storepass eracdleif

# keytool -import -v -trustcacerts -alias Server -file client.cer -
keystore .infoServerKeyStore -keypass eracdleif -storepass eracdleif
```

Now we may delete the exported certificates (the `.cer` files), and all that should remain are two keystore files; `.infoClientKeyStore` and `.infoServerKeyStore`. The server class imports the `.infoServerKeyStore` as a trusted keystore, and the client class imports the `.infoClientKeyStore` as a trusted keystore.

6.6.3 SECURITY POLICY

The InfoServer has enabled the RMI Security Manager. This security manager acts as a reference monitor in Java, and therefore we have to define a security policy for the application. The easiest way to do this is to create a standard text file that contains the security policy, and then set the system property `java.security.policy` to refer to the policy file at run-time. The security policy file we created for the InfoServer is shown in Appendix C.4.3.

When starting the InfoServer, we have to specify the policy file as follows.

```
# java -Djava.security.policy=InfoServerSecurityPolicy -Djava.rmi.
server.hostname=192.168.1.1 Server
```

The `hostname` property is used to clearly specify which network interface that should be bound in the RMI Registry.

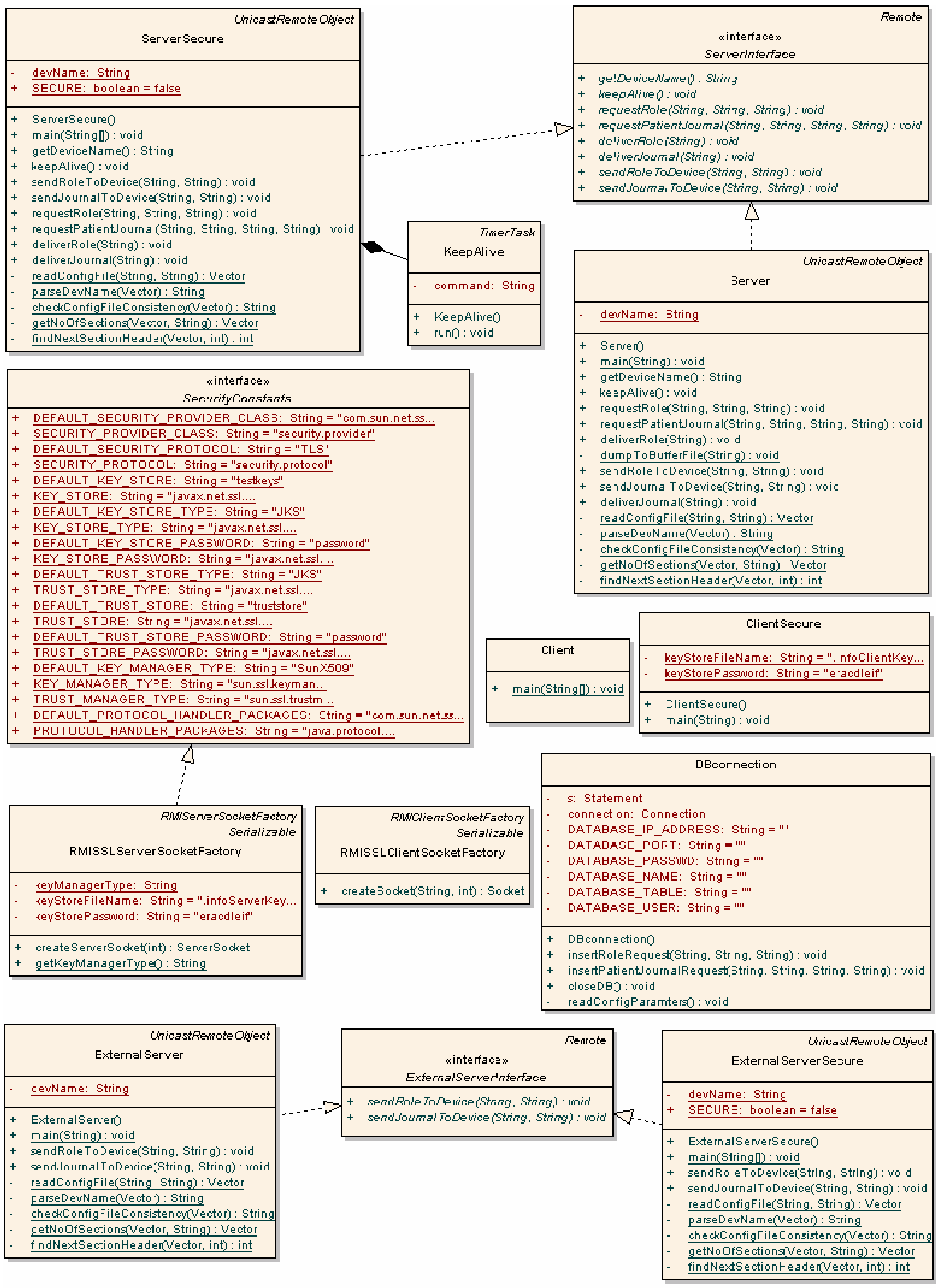
6.6.4 NO SUPPORT FOR JSSE IN BLACKDOWN'S JRE

JSSE is not included in the Java Runtime Environment (JRE) version from Blackdown, which is installed on the MDAs (see Appendix A.5 for more details), and this JRE is unfortunately incompatible with Sun's JSSE for Java 1.3.1. Therefore we were unable to use the secured InfoServer in our FieldCare prototype, but this is only a temporary problem, as newer versions of this JRE will probably include JSSE.

Both InfoServer versions are implemented, but only the InfoServer without SSL will work in a scenario where the MDA devices are a part of the WLAN. The InfoServer files with SSL support are called, `ServerSecure` and `ClientSecure`, while the InfoServer files without SSL support are simply called, `Server` and `Client`.

With the occurrence of this problem, we created a separate security policy file for the standard InfoServer (no SSL support) called `InfoServerSecurityPolicyNoSSL`, although in theory the other policy file should also work.

6.6.5 INFOSERVER CLASS DIAGRAM



UnicastRemoteObject
ServerSecure
- devName: String
+ SECURE: boolean = false
+ ServerSecure()
+ main(String[]) : void
+ getDeviceName() : String
+ keepAlive() : void
+ sendRoleToDevice(String, String) : void
+ sendJournalToDevice(String, String) : void
+ requestRole(String, String, String) : void
+ requestPatientJournal(String, String, String, String) : void
+ deliverRole(String) : void
+ deliverJournal(String) : void
- readConfigFile(String, String) : Vector
- parseDevName(Vector) : String
- checkConfigFileConsistency(Vector) : String
- getNoOfSections(Vector, String) : Vector
- findNextSectionHeader(Vector, int) : int
Remote
«interface»
ServerInterface
+ getDeviceName() : String
+ keepAlive() : void
+ requestRole(String, String, String) : void
+ requestPatientJournal(String, String, String, String) : void
+ deliverRole(String) : void
+ deliverJournal(String) : void
+ sendRoleToDevice(String, String) : void
+ sendJournalToDevice(String, String) : void
TimerTask
KeepAlive
- command: String
+ KeepAlive()
+ run() : void
UnicastRemoteObject
Server
- devName: String
+ Server()
+ main(String) : void
+ getDeviceName() : String
+ keepAlive() : void
+ requestRole(String, String, String) : void
+ requestPatientJournal(String, String, String, String) : void
+ deliverRole(String) : void
- dumpToBufferFile(String) : void
+ sendRoleToDevice(String, String) : void
+ sendJournalToDevice(String, String) : void
+ deliverJournal(String) : void
- readConfigFile(String, String) : Vector
- parseDevName(Vector) : String
- checkConfigFileConsistency(Vector) : String
- getNoOfSections(Vector, String) : Vector
- findNextSectionHeader(Vector, int) : int
«interface»
SecurityConstants
+ DEFAULT_SECURITY_PROVIDER_CLASS: String = "com.sun.net.ss...
+ SECURITY_PROVIDER_CLASS: String = "security.provider"
+ DEFAULT_SECURITY_PROTOCOL: String = "TLS"
+ SECURITY_PROTOCOL: String = "security.protocol"
+ DEFAULT_KEY_STORE: String = "testkeys"
+ KEY_STORE: String = "javax.net.ssl....
+ DEFAULT_KEY_STORE_TYPE: String = "JKS"
+ KEY_STORE_TYPE: String = "javax.net.ssl....
+ DEFAULT_KEY_STORE_PASSWORD: String = "password"
+ KEY_STORE_PASSWORD: String = "javax.net.ssl....
+ DEFAULT_TRUST_STORE_TYPE: String = "JKS"
+ TRUST_STORE_TYPE: String = "javax.net.ssl....
+ DEFAULT_TRUST_STORE: String = "truststore"
+ TRUST_STORE: String = "javax.net.ssl....
+ DEFAULT_TRUST_STORE_PASSWORD: String = "password"
+ TRUST_STORE_PASSWORD: String = "javax.net.ssl....
+ DEFAULT_KEY_MANAGER_TYPE: String = "SunX509"
+ KEY_MANAGER_TYPE: String = "sun.ssl.keyman...
+ TRUST_MANAGER_TYPE: String = "sun.ssl.trustm...
+ DEFAULT_PROTOCOL_HANDLER_PACKAGES: String = "com.sun.net.ss...
+ PROTOCOL_HANDLER_PACKAGES: String = "java.protocol....
Client
+ main(String[]) : void
ClientSecure
- keyStoreFileName: String = ".infoClientKey...
- keyStorePassword: String = "eracdleif"
+ ClientSecure()
+ main(String) : void
RMIServerSocketFactory
Serializable
RMISSLServerSocketFactory
- keyManagerType: String
- keyStoreFileName: String = ".infoServerKey...
- keyStorePassword: String = "eracdleif"
+ createServerSocket(int) : ServerSocket
+ getKeyManagerType() : String
RMIClientSocketFactory
Serializable
RMISSLClientSocketFactory
+ createSocket(String, int) : Socket
DBconnection
- s: Statement
- connection: Connection
- DATABASE_IP_ADDRESS: String = ""
- DATABASE_PORT: String = ""
- DATABASE_PASSWD: String = ""
- DATABASE_NAME: String = ""
- DATABASE_TABLE: String = ""
- DATABASE_USER: String = ""
+ DBconnection()
+ insertRoleRequest(String, String, String) : void
+ insertPatientJournalRequest(String, String, String, String) : void
+ closeDB() : void
- readConfigParamters() : void
UnicastRemoteObject
ExternalServer
- devName: String
+ ExternalServer()
+ main(String) : void
+ sendRoleToDevice(String, String) : void
+ sendJournalToDevice(String, String) : void
- readConfigFile(String, String) : Vector
- parseDevName(Vector) : String
- checkConfigFileConsistency(Vector) : String
- getNoOfSections(Vector, String) : Vector
- findNextSectionHeader(Vector, int) : int
Remote
«interface»
ExternalServerInterface
+ sendRoleToDevice(String, String) : void
+ sendJournalToDevice(String, String) : void
UnicastRemoteObject
ExternalServerSecure
- devName: String
+ SECURE: boolean = false
+ ExternalServerSecure()
+ main(String[]) : void
+ sendRoleToDevice(String, String) : void
+ sendJournalToDevice(String, String) : void
- readConfigFile(String, String) : Vector
- parseDevName(Vector) : String
- checkConfigFileConsistency(Vector) : String
- getNoOfSections(Vector, String) : Vector
- findNextSectionHeader(Vector, int) : int

Figure 6.20 – InfoServer class diagram
An UML based class diagram of the InfoServer system. The standard InfoServer (no SSL support), the secured InfoServer, the standard External InfoServer and the secured External InfoServer are all displayed in this figure together with their interrelationships.

6.6.6 FUTURE RECOMMENDATION

The InfoServer is now implemented as a separate application external to the FieldCare application, and this is not in accordance with how SINTEF envisions the distribution of FieldCare, namely that everything is contained within one single Java Archive (JAR) file. Therefore it should be considered incorporating the InfoServer components into the FieldCare application in the future. It should also be considered to rename the InfoServer system, to avoid the confusion between the server part, client part and the system as a whole. Whether there are an alternative JSSE support for the iPAQs, should also be investigated.

6.7 COMPETENCE ROLE ACQUISITION

As the activity diagram of the Routing Management class (presented in Figure 6.14) shows; the end state is the role acquisition. This is where the device will request a Competence Role from the hospital server.

It is important not to confuse this Competence Role with the device role used on the accident site (medic, coordinator) nor with accident management roles (e.g. Medical Team Leader, Rescue Leader, etc.).

The Competence Role specifies the access level assigned to the person logged in on the device. The access level is used by the hospital server to differentiate the information sent back to the accident site. A level 1 access will provide the most basic information from a patient journal, e.g. the blood type, while a level 4 access will provide highly sensitive data such as mental disorders and sexual diseases. The Competence Role is related to how competent a person is; hence a doctor will have a Competence Role with higher access level than the Competence Role to e.g. a paramedic.

6.7.1 ACTIVITY DIAGRAM

The very first thing we do is checking whether the FCC device is online. This is done by querying the address list, which holds the mapping between the IP address and the associated device name, for the entry "`FCC`". The address list has been previously updated with the newest information provided by OLSR. If the address list contains the element "`FCC`", then this device is online and ready to serve.

The FCControl class holds a couple of static variables specifying various states of the device; `HAS_REQUESTED_ROLE` and `HAS_ACQUIRED_ROLE`. A role request will be issued if the device has not previously requested a role or if the device has already sent a request but has not yet received a reply. When it receives a reply, it will set the Competence Role and enable the *Patient Journal* button. The request and delivery sequences are presented in subsections 6.7.2 and 6.7.3.

The following activity diagram displays the logic in the Routing Management class for the role acquisition, as it was presented in the above paragraphs.

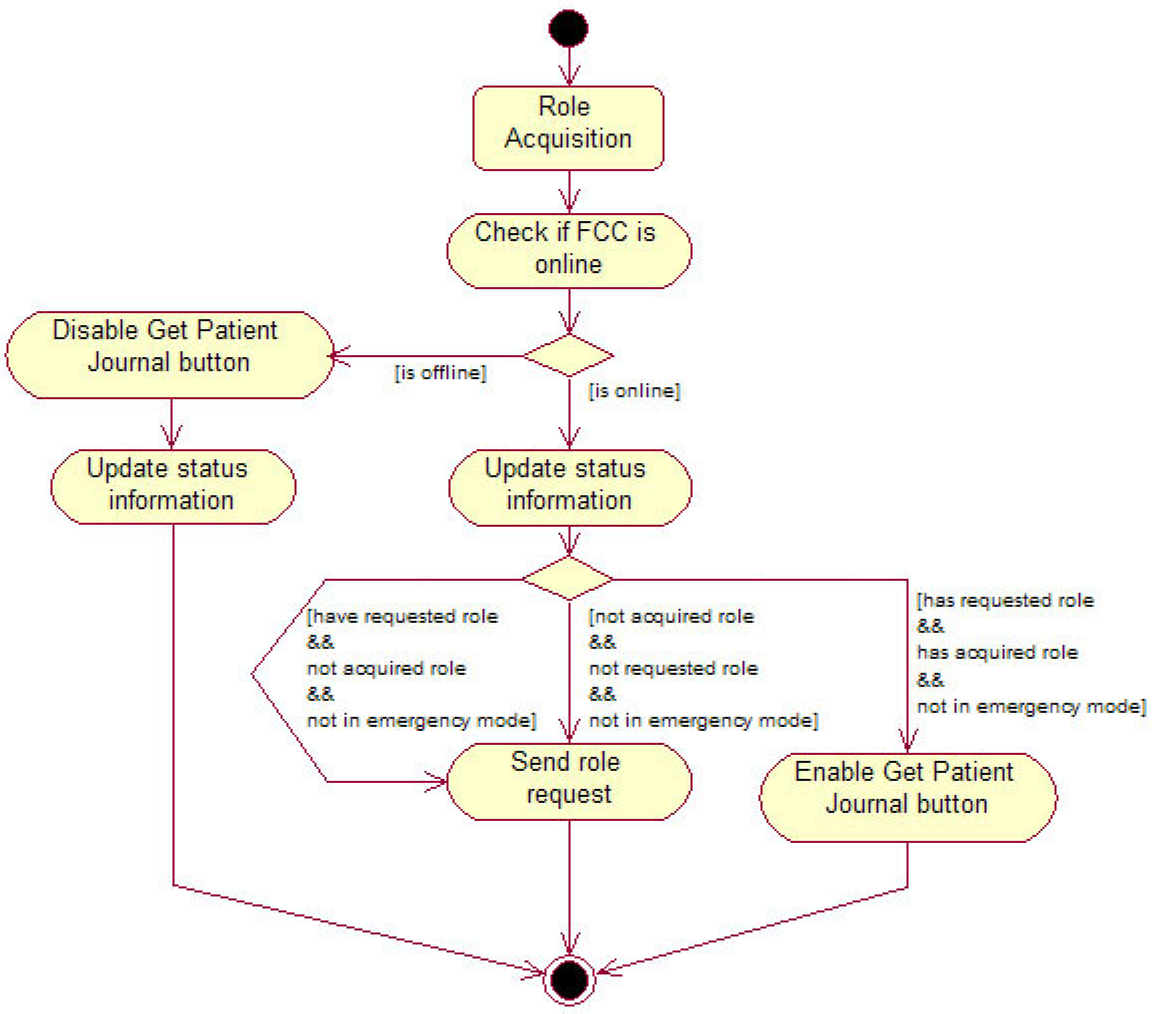

Role Acquisition
Check if FCC is online
Disable Get Patient Journal button
[is offline]
[is online]
Update status information
Update status information
[have requested role && not acquired role && not in emergency mode]
[not acquired role && not requested role && not in emergency mode]
[has requested role && has acquired role && not in emergency mode]
Send role request
Enable Get Patient Journal button

Figure 6.21 – Activity diagram of the Role Acquisition state
This UML activity diagram shows the end state of the Routing Management diagram from Figure 6.14.

6.7.2 ROLE REQUEST

Figure 6.22 below shows the message sequence in a role request. Note that we stop the sequence at the DBconnection class on the FCC device, as this represents the end of what we have focused our work on (see Figure 2.6). The FCCU implementation by Group 2 will handle the request after the FCC has inserted the request into the FCCDB machine.

All message exchanges are done in a push-approach, meaning that when the request has been made, the issuing class does not wait for a reply, allowing the user to go about doing other things.

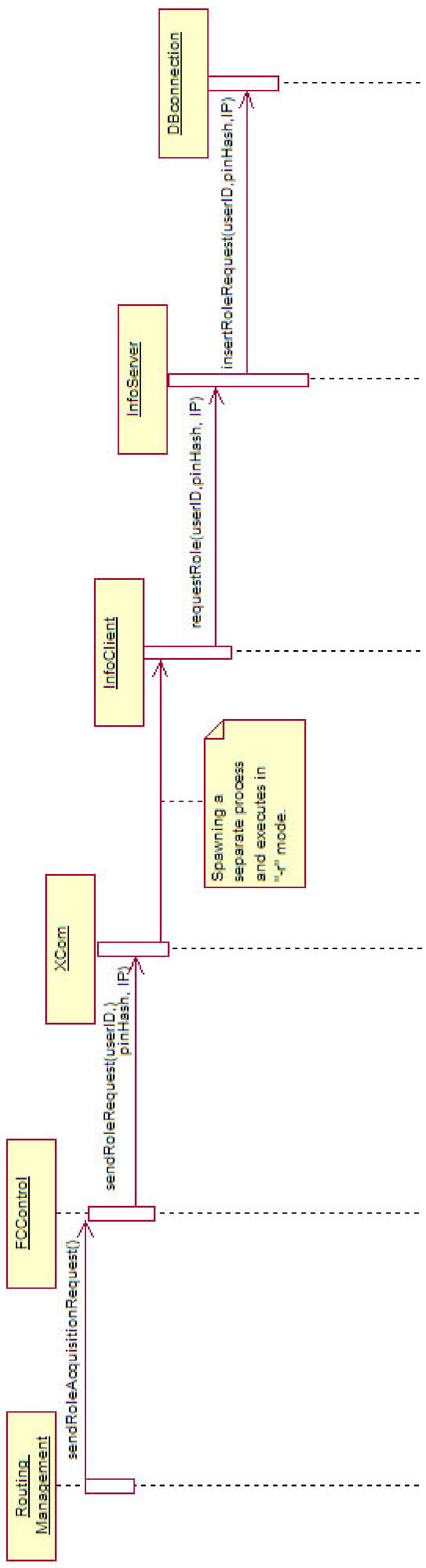

Routing Management
FCControl
XCom
InfoClient
InfoServer
DBconnection
sendRoleAcquisitionRequest()
sendRoleRequest(userID, pinHash, IP)
Spawning a separate process and executes in "-r" mode.
requestRole(userID,pinHash, IP)
insertRoleRequest(userID,pinHash,IP)

Figure 6.22 – Role request sequence

6.7.3 ROLE DELIVERY

The UML sequence diagram presented below shows the sequence of messages when a role is being delivered to the requesting device. Again, notice that we show the sequence from the External InfoServer. This is not the real start of the sequence, but it represents the start of our area of focus. The sequence from the hospital server database to the External InfoServer on the FCC device is the responsibility of Group 2.

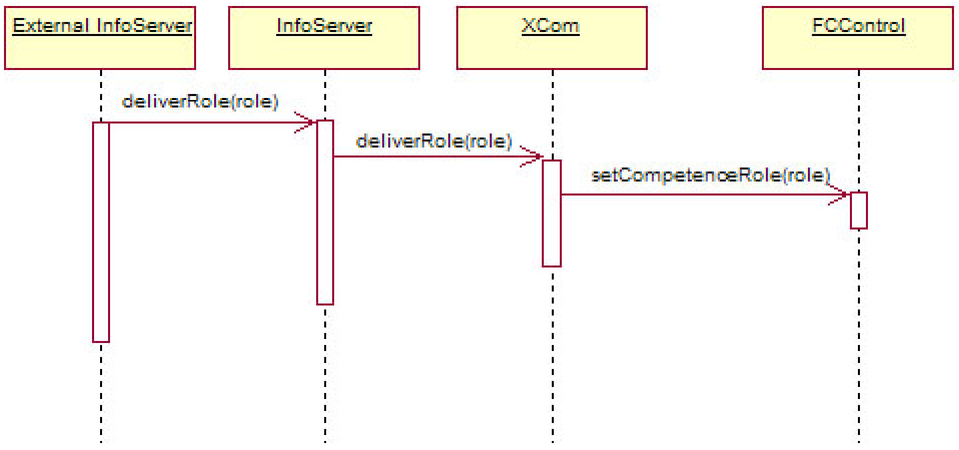


Figure 6.23 – Role delivery sequence

When the FCControl class receives the Competence Role it will enable the *Patient Journal* button, allowing the user to enter a social security number (SSN) in order to transfer a patient journal (as show in Figure 6.24 in section 6.8).

If the FCC device, for some reason, goes offline, the Routing Management class will discover this (through OLSR), and disable the Patient Journal button. However, the FieldCare application will cache the role it received earlier, so when the FCC is online again, it doesn't have to send a new role request, and the Patient Journal button is enabled almost immediately.

6.7.4 FUTURE RECOMMENDATION

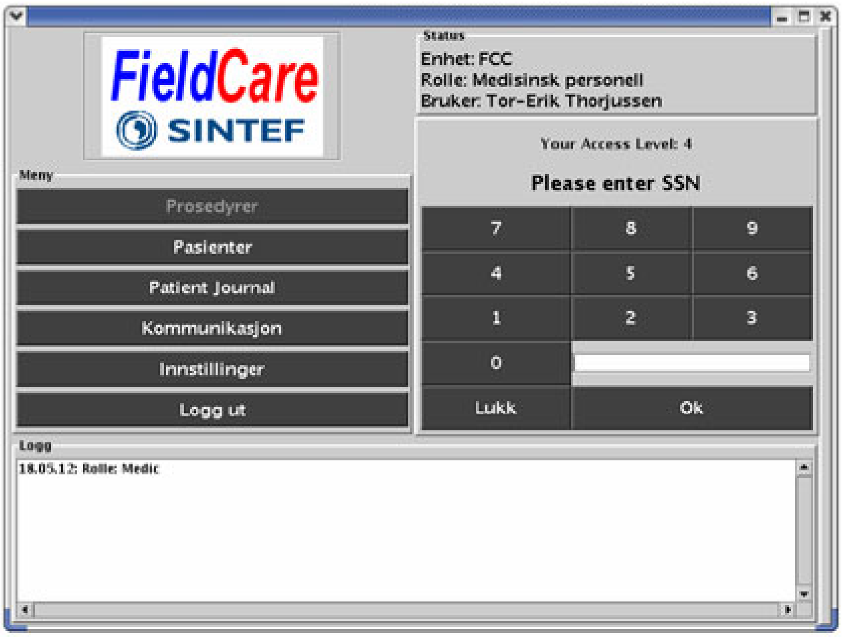
Although Figure 6.23 represents our initial implementation, we realized that with this implementation there exists an incompatibility with the FCControl class. When the XCom class receives a role forwarded by the InfoServer, it has no reference to the instantiated FCControl object, and is therefore unable to further deliver the role. The temporary solution of this problem is to buffer all received data to disk, and then have the FCControl read it accordingly. This is more of a hack than an actual solution, but the lack of time guided our decision in this matter. So in the current implementation, FCControl will start checking a given file (`buffer.file`) after the role request has been issued, and XCom will dump all data to this file when it receives the role. It is therefore recommended that a more elegant solution should be implemented. Of course, if all the elements could be incorporated into the FieldCare application and the use of static methods are avoided, it should not be a problem delivering the data directly.

6.8 PATIENT JOURNAL ACQUISITION

After the device has received a Competence Role, the user is now allowed to issue requests for patient journals, and the information this user will receive is dependent on the access level of the Competence Role.

6.8.1 JOURNAL REQUEST

By pressing the Patient Journal button the user is able to enter an 11-digit social security number, as presented in Figure 6.24.



Status
Enhet: FCC
Rolle: Medisinsk personell
Bruker: Tor-Erik Thorjussen

18.05.12: Rolle: Medic

Figure 6.24 – Screenshot of the SSNPanel
The user has received a Competence Role and is now able to click on the Patient Journal button. This screenshot shows the panel where the user can enter a social security number.

When FCControl class receives a journal request, it will forward it to the XCom class which in turn will use the InfoClient to transfer the request to the InfoServer on the FCC device. The FCC will handle this request in the same way it handled a role request, by inserting the request into the database on the FCCDB device. Now the responsibility of the request has been transferred to Group 2.

The journal request operates in the same way as a role request. This means that the issuing device is not waiting for the journal to be delivered, and allows the user to do other tasks while the rest of the system processes the request.

This sequence of events is shown in detail in the UML sequence diagram below.

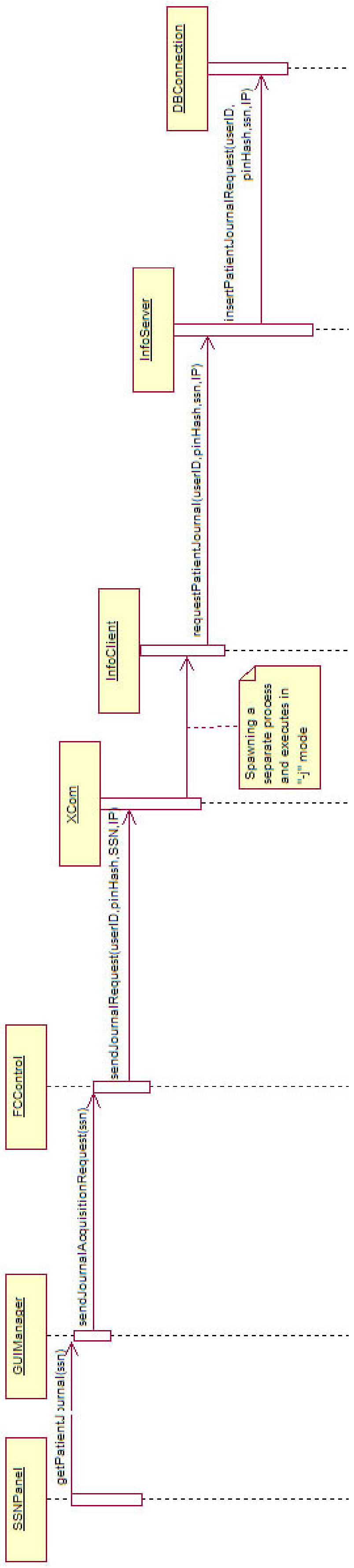

SSNPanel
GUIManager
FCControl
XCom
InfoClient
InfoServer
DBConnection
getPatientJournal(ssn)
sendJournalAcquisitionRequest(ssn)
sendJournalRequest(userID,pinHash,SSN,IP)
Spawning a separate process and executes in "-j" mode
requestPatientJournal(userID,pinHash,ssn,IP)
insertPatientJournalRequest(userID, pinHash,ssn,IP)

Figure 6.25 – Patient journal request sequence

6.8.2 JOURNAL DELIVERY

The journal request contains among other things the ID of the user requesting the journal and a hashed value of this user's PIN. By using this information, the hospital server is able to check the Competence Role of this user, and do an information filtering based on the access level associated with the role. The filtering is done at the hospital server for security reasons (to minimize the amount of sensitive data transferred to the external accident site). However this processing is carried out by Group 2's implementation and our domain begins first at the External InfoServer on the FCC. This is also the basis for the UML sequence diagram shown in Figure 6.26, i.e. we start our sequence when the External InfoServer receives data from the hospital server.

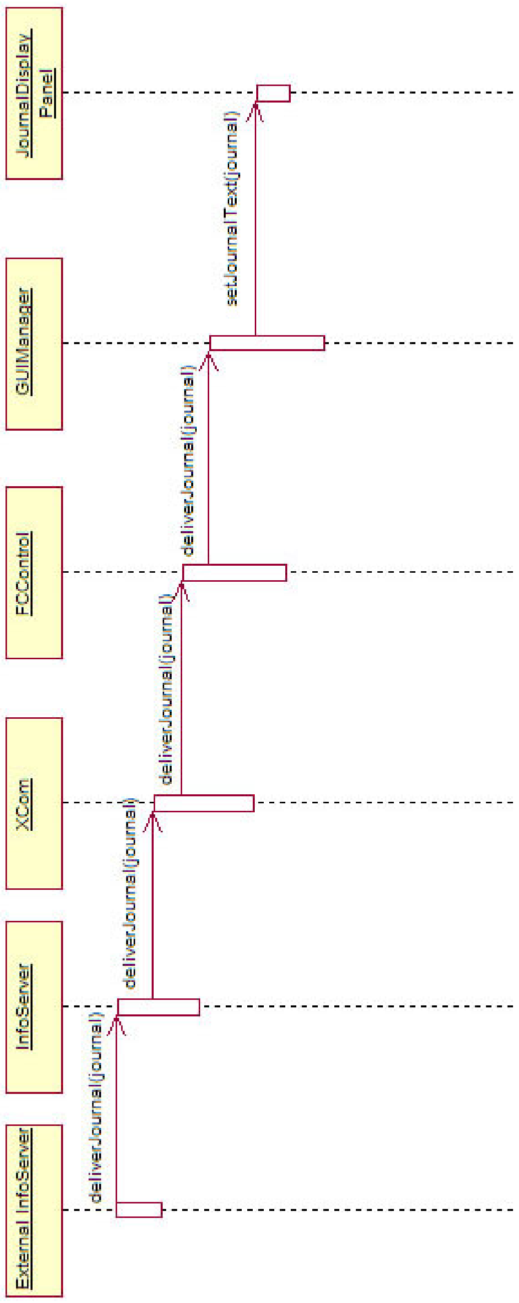
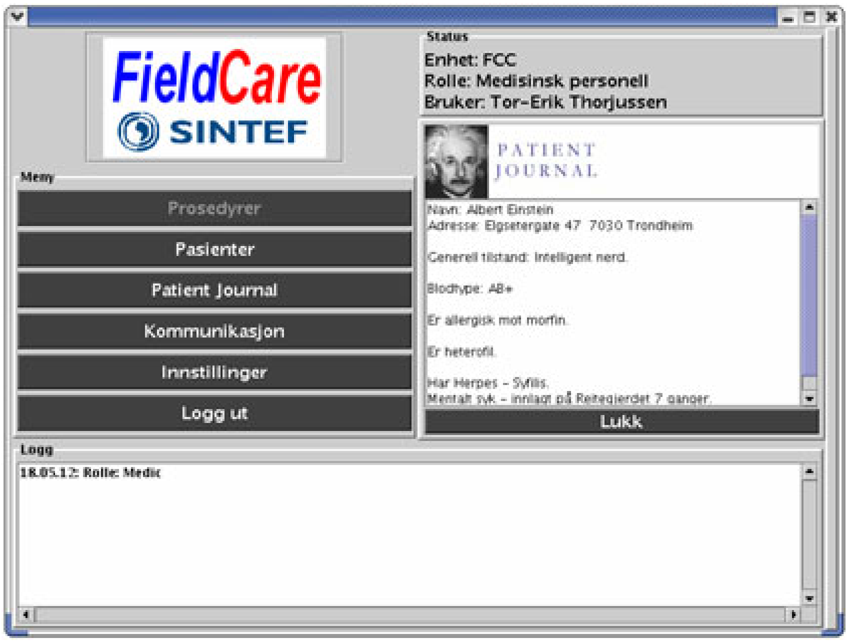


Figure 6.26 – Patient journal delivery sequence

When the JournalDisplayPanel class finally receives the journal, it will present the data to the user in a pop-up panel. A screenshot of this panel is shown in Figure 6.27.



Enhet: FCC
Rolle: Medisinsk personell
Bruker: Tor-Erik Thorjussen

Navn: Albert Einstein
Adresse: Elgsetergate 47 7030 Trondheim

Generell tilstand: Intelligent nerd.

Blodtype: AB+

Er allergisk mot morfin.

Er heterofil.

Har Herpes - Syfilis.
Mentalt syk - innlagt på Reitegjerdet 7 ganger.

18.05.12: Rolle: Medic

Figure 6.27 – Displaying a patient journal
This is an example of a patient journal being displayed.

6.8.3 FUTURE RECOMMENDATION

Since the journal delivery sequence is basically the same as for the role delivery, this implementation also suffers of the problem stated in 6.7.4. So the journals received by the XCom class are also buffered to a file, which the FCControl class then reads. Although FCControl will delete the buffer file when it has been read, this process leaves the contents of the journal open for five seconds (in the worst case), and this constitutes a major security vulnerability. However, as we also stated in 6.7.4, this solution is more of a hack than an actual solution, and the same recommendations as given for the role request apply.

6.9 SUMMARY

The implementation work carried out in this project, together with the work of Group 2, has created a new FieldCare prototype with the following features added; proper access control and authentication using iButtons, dynamic routing, the possibility to securely download patient journals to devices used on the accident site, and information filtering based on the access level of each user. Our specific recommendations and comments to each aspect of our implementation are best understood by reading the *future recommendation* sections above.

7 CONCLUSION

We have defined and analyzed security issues in a medical emergency context, and we have implemented some of the defined security requirements. The setting and scenario we have considered has been based on the FieldCare prototype developed by SINTEF Telecom and Informatics. We have extended the functionality of this prototype with several new features; access control with iButtons, secure journal retrieval and dynamic routing.

We have used the ISO/IEC TR 13335 and the IEC 300-3-9 standards to categorize and evaluate the identified security threats and risks associated with the FieldCare system. Denial of Service attacks and incorrect routing was both assigned the highest risk level, because these attacks had the potential of completely paralyzing the system. The standards provided us with a formal guideline, however it should be noted that the assignment of the hazard risk assessment code to each individual threat was still a selective process, dependent on our level of experience.

The Data Inspectorate verified our interpretations of the regulations applicable to the FieldCare system. They stated, among other things, the necessity of an end-to-end encryption when dealing with sensitive information. This requirement was taken into consideration in our implementation, where we enabled the use of SSL encryption in order to be in more compliance with the regulations provided by the Data Inspectorate.

We have introduced an electronic identity token called PiTag containing user information and PIN code. To realize the PiTag we have used the iButton technology, which satisfied our durability requirement. The PiTag was used in the implementation of a new and improved access control for the logon procedure. To secure the user's PIN code, we used the SHA-1 hash algorithm to make a message digest of the code. The SHA-1 algorithm has been FIPS approved by NIST. The SHA-1 algorithm was chosen because it has no known flaws and was easy to implement using the standard Java API.

Since the Emergency Login is necessary to allow any persons to assist in medical emergency situations, we have defined an access restriction in our security requirement specification. This requirement has been satisfied in our new system prototype, by using Competence Roles. The concept of Competence Roles is to give only authenticated users access to the external network and the possibility to download patient journals. This Competence Role is not given to users logged on in Emergency mode. The Emergency Login is now in compliance with the requirements given by the Data Inspectorate.

To improve the management, scalability and reliability of the ad-hoc wireless network on the site of the accident, we implemented the OLSR routing protocol on the mobile devices. The performance of the proactive OLSR protocol was tested and the results of the test proved to be quite adequate for a prototype implementation. However, for a future FieldCare implementation we recommend the SAODV protocol. This protocol is both reactive and provides a much greater security of the routing control data.

LIST OF ACRONYMS

ABR	Associativity-Based Routing
ACL	Access Control List
ACU	Aironet Client Utility (Software developed by Cisco)
AES	Advanced Encryption Standard
AMK	Akuttmedisinsk Kommunikasjonssentral (Eng: EMCC)
AODV	Ad-Hoc On-Demand Distance Vector
AP	Access Point
API	Application Programming Interface
b	Bit
B	Byte (8 bits)
CBAC	Context Based Access Control
CCK	Complementary Code Keying
CD	Compact Disc
CDAC	Content Dependent Access Control
CECOM	Communication-Electronics Command
CEDAR	Core-Extraction Distributed Ad-Hoc Routing
CMDA	Coordinators Medical Assistant
COTS	Commercial Off-The-Shelf
CRC	Cyclic Redundancy Code
CRL	Cambridge Research Lab
CRUDE	Collector for RUDE
DAC	Discretionary Access Control
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed Denial of Service
DES	Data Encryption Standard
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Server
DoS	Denial of Service
DREC	Dynamic Reciever (Application software designed by NRL)
EAP	Extensible Authentication Protocol
EEPROM	Electrically Eraseable Programmable Read-Only Memory
EMCC	Emergency Medical Communication Centre (Nor: AMK)
EMI	Electromagnetic Interference
EMP	Electromagnetic Pulse
ESP	Encapsulating Security Payload
ESSID	Extended Service Set Identifier
FCC	FieldCare Connector
FCCDB	FieldCare Connector Database
FCCU	FieldCare Connector Update
FGU	Fast Graphical Update
FIPS	Federal Information Processing Standard
GSA	General Services Administration
GUI	Graphical User Interface
HP	Hewlett-Packard
HSDB	Hospital Server Database
Hz	Hertz
IBAC	Identity Based Access Control
IEEE	Institute of Electrical & Electronics Engineers
IETF	Internet Engineering Task Force
IBA	iButton Admin

IBM	International Business Machines Corporation
IC	Integrity Check
ICT	Information and Communication Technology
IEC	International Electrotechnical Commission
INRIA	The French National Institute for Research in Computer Science and Control
IME	Faculty of Information Technology, Mathematics and Electrical Engineering
IP	Internet Protocol
IPSec	IP Security
IS	Information Security
ISBN	International Standard Book Number
ISO	International Organization for Standardization
ITEM	Department of Telematics
IV	Initialization Vector
JAR	Java Archive
JRE	Java Runtime Environment
JSSE	Java Secure Socket Extension
K	Kilo (10^4)
KoKom	National Centre on Emergency Health-Care Communication
LAN	Local Area Network
LEAP	Lightweight Extensible Authentication Protocol
M	Mega (10^6)
MAC	Mandatory Access Control
MAC	Media Access Control
MANET	Mobile Ad-hoc Network
MD4	Message Digest Algorithm version 4
MD5	Message Digest Algorithm version 5
MDA	Medical Digital Assistant
MGEN	Multi-Generator (Application software designed by NRL)
MIT	Massachusetts Institute of Technology
MiTag	Medical Information Tag
MOS	Metal Oxide Semiconductor
MPR	Multi-point relay (used in the OLSR protocol)
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
NRL	Naval Research Laboratory
NSA	National Security Agency
NTNU	The Norwegian University of Science and Technology
NV	Non-Volatile
ODMRP	On-Demand Multicast Routing Protocol
OLSR	Optimized Link State Routing
OS	Operating System
OSS	Open Source Software
P2P	Peer-to-Peer
PBAC	Policy Based Access Control
PC	Personal Computer
PCMCIA	Personal Computer Memory Card International Association
PiTag	Personal Information Tag
PROTEAN	Protocol Engineering Advanced Networking Research Group
ps	Per Second
QPSK	Quaternary Phase Shift Keying
RAC	Risk Assessment Code
RACE	Research in Advanced Communications for Europe
RAM	Random Access Memory
RBAC	Role Based Access Control
RC4	Ron’s Code 4 (Encryption algorithm designed by Ron Rivest)

RF	Radio Frequency
RIPE	RACE Integrity Primitives Evaluation
RIPEMD	RIPE’s Message Digest algorithm
RMI	Remote Method Invocation
RMS	Role Management System
ROM	Read Only Memory
RSBAC	Rule Set Based Access Control
RUDE	Real-time UDP Data Emitter
SAODV	Secure Ad hoc On-Demand Distance Vector Routing
SCP	Secure Copy
SHA	Secure Hash Algorithm
SINTEF	The Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology
SSID	Service Set Identifier
SSH	Secure Shell
SSL	Secure Sockets Layer
SSN	Social Security Number
TBRPF	Topology Dissemination Based on Reverse-Path Forwarding
TC	Topology Control (type of message used by the OLSR protocol)
TORA	Temporally Ordered Routing Algorithm
UBAC	User Based Access Control
UDP	User Datagram Protocol
UML	Unified Modeling Language
URL	Uniform Resource Locator
UPS	Uninterruptible Power Supply
USB	Universal Serial Bus
VBAC	View Based Access Control
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network
XOR	Exclusive Or
ZRP	Zone Routing Protocol

REFERENCES

[ADTI02] Alexis de Tocqueville Institution, "*Opening the Open Source Debate*", June 2002, URL last used: May 2003.
http://www.adti.net/html_files/defense/need%20to%20purchase%20opensource%20white%20paper.pdf

[AG03] Adamson, B., Greenwald, H., "*MGEN - The Multi-Generator Toolset*", Naval Research Laboratory, Protocol Engineering Advanced Networking Research Group, URL last used: May 2003.
http://mgen.pf.itd.nrl.navy.mil/

[BCG01] Bruno, R., Conti, M., Gregori, E., "*WLAN Technologies for Mobile Ad-Hoc Networks*", Consiglio Nazionale delle Ricerche Istituto CNUCE, Italy, 2001, URL last used: May 2003.
http://www.cs.ucsd.edu/classes/wi02/cse294/00927196.pdf

[BGD01] Borisov, N., Goldberg, I., Wagner, D., "*Security of the WEP algorithm*", University of California at Berkeley, 2001, URL last used: May 2003.
http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html

[Bou03] Boucher, P., "*Entrust Cryptographic Module 6.0 - FIPS 140-1 Validation Security Policy*", Computer Security Resource Center, NIST, May 2003, URL last used: May 2003.
http://www.csrc.nist.gov/cryptval/140-1/140sp/140sp176.pdf

[BVA$^+$03] Blunk, L., Vollbrecht, J., Aboba, B., Carlson, J., "*Extensible Authentication Protocol (EAP)*", IETF Internet Draft, May 2003, URL last used: May 2003
http://www.ietf.org/internet-drafts/draft-ietf-eap-rfc2284bis-03.txt

[Cam01] Camelot USA, "*Differentiating Between Access Control Terms*", WindowsSecurity.com, October 2001, URL last used: June 2003.
http://www.windowsecurity.com/uplarticle/2/Access_Control_WP.pdf

[Chr02] Christian, A., "*iPaq H3100/H3600/H3700/H3800 Handhelds.org Bootloader Installation Instructions*", Handhelds.org, 2002, URL last used: May 2003.
http://familiar.handhelds.org/releases/v0.6/install/install-bootldr.html

[CJ03] Clausen, T., Jacquet, P., "*Optimized Link State Routing Protocol*", IETF Internet Draft version 10, May 2003, URL last used: May 2003.
http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-10.txt

[Cla99] Clark, T., "*Network Associates Adds Linux Product*", CNET News, February 1999.

[Col01] Cole, E., "*Hackers Beware*", New Riders, Indianapolis, Indiana 46290, USA, ISBN: 0-7357-1009-0

[Com00] Compaq Computer Corporation, "*Compaq iPAQ H3600 Hardware Design Spesification – Version 0.2f*", May 2000, URL last used: May 2003.
http://www.handhelds.org/Compaq/iPAQH3600/iPAQ_H3600.html

[CS02] Cisco Systems, Inc., "*Release Notes for Cisco Aironet Client Utilities Version 2.0.x for Linux*", URL last used: May 2003.
http://www.cisco.com/en/US/products/hw/wireless/ps4555/prod_release_note09186a008007f795.html

[Dai00] Dai, W., "*Crypto++ 4.0 Benchmarks*", 2000, URL last used: May 2003.
http://www.eskimo.com/~weidai/benchmarks.html

[DBP99] Dobbertin, H., Bosselaers, A., Preneel, B., "*The hash function RIPEMD-160*", Anton Bosselaers, August 1999, URL last used: May 2003.
http://www.esat.kuleuven.ac.be/~bosselae/ripemd160.html

[Del02] Delio, M., "*Did MS Pay for Open-Source Scare?*", Wired News, June 5, 2003. URL last used: June 2003.
http://www.wired.com/news/linux/0,1411,52973,00.html

[DGN+99] DARPA, GSA, NIST and NSA, "*Report on Open Source Code and the Security of Federal Systems*", The Office of the National Coordinator for Security, Infrastructure Protection and Counter-Terrorism, June 1999.

[DSC97] Dallas Semiconductor Corp., "*Book of iButton Standards*", August 1997, URL last used: May 2003. *http://www.ibutton.com/ibuttons/standard.pdf*

[Eat02] Eaton, D., "*Diving into the 802.11i Spec: A Tutorial*", CommsDesign, 2002, URL last used: May 2003. *http://www.commsdesign.com/story/OEG20021126S0003*

[FMS02] Fluhrer, S., Mantin, I., Shamir, A., "*Attacks On RC4 and WEP*", CryptoBytes volume 5 no. 2, 2002, URL last used: May 2003. *http://www.rsasecurity.com/rsalabs/cryptobytes/cryptobytes_v5n2.pdf*

[Gra01] Granger, S., "*Social Engineering Fundamentals, Part I: Hacker Tactics*", SecurityFocus, 2001, URL last used: May 2003. *http://www.securityfocus.com/infocus/1527*

[GSW03] Gorman, J., Svagård, I., Walderhaug, S., "*FieldCare – Information Gathering and Distribution in Emergency Care*", SINTEF Telecom and Informatics, 2002.

[Har97] Harl, "*People Hacking: The Psychology of Social Engineering*", Access All Areas III, Packet Storm, URL last used: May 2003. *http://packetstormsecurity.nl/docs/social-engineering/aaatalk.html*

[HMS03] Henriksen, D., Melby, I., Skjellerudsveen, E., " *Analyse av sikkerheten i et akuttmedisinsk scenario og implementasjon av en sikker kommunikasjons-link over et trådløst medie*", Dept. of Telematics, NTNU, 2003.

[HN02] Hicks, J., Nelson, R., "*Familiar v0.6 Installation Instructions*", Handhelds.org, 2002, URL last used: May 2003. *http://familiar.handhelds.org/releases/v0.6/install/install.html*

[Hor02] Horak, R., "*Wireless LANs (WLANs): Focus on 802.11b*", CommWeb, 2002, URL last used: May 2003. *http://www.commweb.com/article/COM20020827S0003*

[HPJ03] Hu, Y., Perrig, A., Johnson, D.B., "*Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks*", INFOCOM 2003, URL last used: May 2003. *http://www.monarch.cs.rice.edu/monarch-papers/infocom03.pdf*

[HRA01] The Health Registry Act, Norwegian Ministry of Justice, 2001, URL last used: May 2003. *http://www.lovdata.no/all/nl-20010518-024.html*

[Jar03] Jarvi, K., "*RXTX: serial and parallel I/O libraries supporting Sun's CommAPI*", www.rxtx.org, 2003, URL last used: June 2003. *http://www.rxtx.org*

[JMH03] Johnson, D. B., Maltz, D. A., Hu, Y., "*The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*", IETF Internet Draft version 9, April 2003, URL last used: May 2003. *http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-09.txt*

[Kar00] Kärpijoki, V., "*Security in Ad Hoc Network*", Helsinki University of Technology – Telecommunication Software and Multimedia Laboratory, URL last used: February 2003. *http://www.tcm.hut.fi/Opinnot/Tik-110.501/2000/papers/karpijoki.pdf*

[Karu02] Karunanidhi, A., "*Linux on IPAQ in 21 Easy Steps*", Language and Media Processing Laboratory, University of Maryland, 2002, URL last used: May 2003. *http://www.cfar.umd.edu/~arvind/linuxonipaq.htm*

[Ken01] Kenwood, C.A, "*A Business Case Study of Open Source Software*", The MITRE Corporation, July 2001, URL last used: May 2003. *http://www.mitre.org/work/tech_papers/tech_papers_01/kenwood_software/kenwood_software.pdf*

[Kin00] Kingpin, "*A Practical Introduction to the Dallas Semiconductor iButton*",@Stake, Inc., 196 Broadway, Cambridge, MA 02139, USA, August 2000, URL last used: May 2003.
http://www.atstake.com/research/reports/acrobat/practical_introduction_to_ibutton.pdf

[KJJ03] Kocher, P., Jaffe, J., Jun, B., "*Q&A on Differential Power Analysis*", Cryptography Research, Inc., 2003, URL last used: June 2003.
http://www.cryptography.com/resources/whitepapers/DPA-qa.html

[KK99] Köemmerling, O. , Kuhn, M. G., "*Design Principles for Tamper-Resistant Smartcard Processors*", USENIX Workshop on Smartcard Technology (Smartcard '99), Chicago, Illinois, USA, May 1999, URL last used: May 2003.
http://www.cl.cam.ac.uk/~mgk25/sc99-tamper.pdf

[Kop93] Kopp, C., "*A Doctrine for the Use of ElectroMagnetic Pulse Bombs*", Revised draft of RAAF APSC Working Paper #15, 1993, URL last used: May 2003.
http://www.csse.monash.edu.au/~carlo/archive/MILITARY/APSC/wp15-draft.pdf

[LM03] Lee, R., Macker, J., "*OLSRD HOWTO, Version 1.0*", US Naval Research Laboratory, 2003, URL last used: May 2003.
http://downloads.pf.itd.nrl.navy.mil/olsr/olsrdhowto.pdf

[LSP02] Laine, J., Saaristo, S., Prior, R., "*RUDE & CRUDE version 0.7*", SourceForge.net 2002, URL last used: May 2003.
http://rude.sourceforge.net

[Mac03] Macker, J., Reply to Berner Vegge [VGE03] regarding OLSR and AODV test results, IETF Working Group Mailing Lists, URL last used: May 2003.
http://www1.ietf.org/mail-archive/working-groups/manet/current/ msg01703.html

[Mil02] Milner, M., "*Network Stumbler*", 2002, URL last used: May 2003.
http://www.netstumbler.com

[NIST93] National Institute of Standards and Technology, "*FIPS PUB 180 – Secure Hash Standard*", May 1993, URL last used: May 2003.
http://www.itl.nist.gov/lab/fips/fips180.txt

[NIST95] National Institute of Standards and Technology, "*FIPS PUB 180-1 – Secure Hash Standard*", April 1995, URL last used: May 2003.
http://csrc.nist.gov/publications/fips/fips180-1/fip180-1.pdf

[NMS+03] Newman, R.E., Moskowitz, I.S., Syverson, P., Serjantov, A., "*Metrics for Traffic Analysis Prevention*", Workshop on Privacy Enhancing Technologies, March 2003, URL last used: May 2003.
http://petworkshop.org/2003/preproc/04-preproc.pdf

[OLT03] Ogier, R., Lewis, M., Templin, F., "*Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)*", IETF Internet Draft version 8, April 2003, URL last used: May 2003.
http://www.ietf.org/internet-drafts/draft-ietf-manet-tbrpf-08.txt

[OMG03] Open Management Group, Inc., "*OMG Unified Modeling Language Specification, version 1.5*", March 2003, URL last used: May 2003.
http://www.omg.org/docs/formal/03-03-01.pdf

[PBD03] Perkins, C. E., Belding-Royer, E. M., Das, S.R., "*Ad hoc On-Demand Distance Vector (AODV) Routing*", IETF Internet Draft version 13, February 2003, URL last used: May 2003.
http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-13.txt

[PDA00] The Personal Data Act of 2000, The Data Inspectorate, URL last used: May 2003.
http://www.datatilsynet.no/lov/loven/poleng.html

[PL03] Plakoo, A., Laouiti, A., "*OLSR, Optimized Link State Routing*", INRIA, 2003, URL last used: May 2003.
http://menetou.inria.fr/olsr/

[Pri03] Prikryl, M., "*WinSCP 2.3.0*", April 2003, URL last used: May 2003.
http://winscp.vse.cz/eng/index.php

[Riv92] Rivest, R. L., "*Request for Comments: 1321 - The MD5 Message-Digest Algorithm*", MIT Laboratory for Computer Science and RSA Data Security, Inc., April 1992, URL last used: May 2003.
http://www.rfc-editor.org/rfc/rfc1321.txt

[Roy99] Royer, E.M, "*A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks*", IEEE Personal Communications, April 1999, URL last used: May 2003.
http://ntrg.cs.tcd.ie/htewari/papers/royer.pdf

[RSA00] RSA Laboratories, "*RSA Laboratories' Frequently Asked Questions About Today's Cryptography, Version 4.1*", RSA Security Inc., 2000, URL last used: May 2003.
http://www.rsasecurity.com/rsalabs/faq/index.html

[SAZ02] Security Advisor Zone, "*Who Do You Trust: Open Source or Proprietary Code?*", Doc # 10285, August 2002, URL last used: May 2003.
http://securityadvisor.info/Articles.nsf/dp/7CD2BB6B1D351B3F88256BFB00810B27

[SB02] Singh, J. P., Bambos, N., "*Proposal and Demonstration of Link Connectivity Assessment based Enhancements to Routing in Mobile Ad-hoc Networks*", NetLab, Stanford University, 2002, URL last used: February 2003.
http://www.stanford.edu/~jatinder/academic/projects/ongoing/adhoc_routing/adhoc_routing.htm

[Sch95] Schneier, B., "*Applied Cryptography, second edition*", John Wiley & Sons, 1995, ISBN: 0471117099

[SE03] Swaminatha, T.M., Elden, C.R, "*Wireless Security and Privacy – Best Practices and Design Techniques*", Addison-Wesley, Pearson Education Inc., Boston, MA 02116, USA, ISBN: 0-201-76034-7

[SHB03] Snax (real name unknown), Hegerle, B., Bruestle, J., "*AirSnort – Homepage*", URL last used: May 2003.
http://airsnort.shmoo.com

[SIN02] SINTEF, "*FieldCare: Integrated Support for Dealing with Medical Emergencies*", White Paper, June 2002.

[Sko02] Skoll, D., "*Opening the Open-Source Debate*", Roaring Penguin Software, Inc., 2002, URL last used: June 2003.
http://www.roaringpenguin.com/adti2.php3

[SS03] SearchSecurity.com, "*Acronyms & Term Definitions*", URL last used: May 2003.
http://searchsecurity.techtarget.com/glossary/0,294242,sid14,00.html

[Sta00] Ståhlberg, M., "*Radio Jamming Attacks Against Two Popular Mobile Networks*", Helsinki University of Technology, 2000, URL last used:
http://www.tcm.hut.fi/Opinnot/Tik-110.501/2000/papers/stahlberg.pdf

[Sta98] Stallings, W., "*Cryptography and Network Security – Principles and Practice, Second Edition*", Prentice Hall, Upper Saddle River, New Jersey 07458, ISBN: 0-13-869017-0

[Stu01] Sturgeon, A. (Project Editor), "*ISO/IEC TR 13335 – Information Technology – Security Techniques*", Secretariat ISO/IEC JTC 1/SC 27, April 2001.

[Sun01] Sun Microsystems, Inc., "*Keytool – Key and Certificate Management Tool*", URL last used: April 2003.
http://java.sun.com/products/jdk/1.2/docs/tooldocs/solaris/keytool.html

[Sun03a] Sun Microsystems, Inc., "*Java Card Technology*", 2003, URL last used: June 2003.
http://java.sun.com/products/javacard/

[Sun03b] Sun Microsystems, Inc., "*Java Secure Socket Extension (JSSE) 1.0.3_01*", URL last used: May 2003.
http://java.sun.com/products/jsse/index-103.html

[SV100] SV-100:2000, "*Sikkerhetsbestemmelsene i personalopplysningsforskriften med kommentar*", The Data Inspectorate, 2000, URL last used: May 2003.
http://www.datatilsynet.no/dtweb/attachment/777/SV100_00.pdf

[T1A101] T1A1, Technical Subcommittee on Performance and Signal Processing, "*American National Standard for Telecommunications - Telecom Glossary 2000 – T1.523-2001*", American National Standards Institute, Inc., 2001, URL last used: May 2003.
http://www.atis.org/tg2k/t1g2k.html

[TDI03] The Data Inspectorate (Datatilsynet), "*General Information in English*", 2003, URL last used: May 2003.
http://www.datatilsynet.no/eng.html

[Tho03] Thorjussen, T., "*Writing to file problem*", 1-Wire Software Development – Online Discussion Group, April 2003, URL last used: June 2003.
http://lists.dalsemi.com/maillists/1-wire-software-development/2003-April/001483.html

[TL01] Torgerson, M., Leeuwen, B.V., "*Routing Data Authentication in Wireless Ad Hoc Networks*", Sandia National Laboratories, USA, October 2001, URL last used: May 2003.
http://infoserve.sandia.gov/cgi-bin/techlib/access-control.pl/2001/ 013119.pdf

[USA02] U.S. Army, Headquarters, Department of the Army, "*TM 5-690, Grounding and Bonding in Command, Control, Communications, Computer, Intelligence, Surveillance, and Reconnaissance (C4ISR) Facilities, Chapter 5*", February 2002, URL last used: May 2003.
http://www.usace.army.mil/inet/usace-docs/armytm/tm5-690/entire.pdf

[USA99] U.S. Army Communication-Electronics Command, Directorate for Safety, "*CECOM Risk Management Matrix*", CECOM, Fort Monmouth, NJ, USA, URL last used: May 2003.
http://www.monmouth.army.mil/cecom/safety/system/spub.htm

[Veg03] Vegge, B., "*Ad-Hoc Routing Protocols – Test Report*", Applica AS, Munin-Applica Document 8 Edition 2, April 2003.

[VDJ+03] Vos, J., Dost, E.C., Jong, K., Sinz, M., Reilly, P.M., Groot, C., Hutinger, S., Matola, T., Kreileder, J., Asha, K., Wynne, S., "*The Blackdown Project*", URL last used: May 2003.
http://www.blackdown.org

[VGE03] Vegge, B., Gyland, G., Egeland, G.I., "*Mobile Ad Hoc NETworks*", Nettverk & Kommuniksjon no. 3 2003, IDG Norge AS.

[Was02] Wassenberg, W., "*Java Comm Serial API How-To for Linux*", wass.homelinux.net, December 2002, URL last used: June 2003.
http://wass.homelinux.net/howtos/Comm_How-To.shtml

[WDM+03] Weston, J., Dean. J., Macker, J., Lee, R., Chao, W., "*US Naval Research Laboratory – OLSR Project*", NRL, 2003, URL last used: May 2003.
http://pf.itd.nrl.navy.mil/projects/olsr/

[Wil01] Williams, J., "*The IEEE 802.11b Security Problem, Part 1*", IT Professional, November/December 2001, URL last used: January 2003.
http://dlib.computer.org/it/books/it2001/pdf/f6096.pdf

[Zap01] Zapata, M.G. "*Secure Ad hoc On-Demand Distance Vector Routing*", IETF Internet Draft, 2001, URL last used: May 2003.
http://www.ietf.org/internet-drafts/draft-guerrero-manet-saodv-00.txt

[Zap02] Zapata, M.G., "*Secure Ad hoc On-Demand Distance Vector Routing*", Mobile Computing and Commuications Review, volume 6, number 3, July 2002, URL last used: May 2003.
http://delivery.acm.org/10.1145/590000/581312/p106-zapata.pdf?key1=581312&key2=7683153501&coll=ACM&dl=ACM&CFID=10515221&CFTOKEN=11719758

[Zat96] Zatko, P. M., "*L0pht Security Advisory*", @Stake, Inc., 196 Broadway, Cambridge, MA 02139, USA, Nov 1996, URL last used: May 2003.
http://www.atstake.com/research/advisories/1996/krb_adv.txt

APPENDIX A

All installation and configuration guides of the software used in this project are located in this appendix.

A.1 INSTALLATION OF LINUX FAMILIAR 0.6 ON IPAQ H3660

This installation is in part based on [Karu02], [HN02] and [Chr02], but also our own experiences.

A.1.1 REQUIREMENTS

The hardware requirements are as following:

- HP (Compaq) iPAQ H3660
- HP (Compaq) AutoSync serial cable or serial cradle
- HP (Compaq) USB cradle
- HP (Compaq) jacket (includes 2xPCMCIA slots and extra battery)
- Microsoft Windows 98/ME/NT/2000/XP
- PC with a serial port
- Preferably a PCMCIA WLAN card

As for the software required:

On your Windows machine you need to install the following:

- Microsoft Active Sync 3.5 (The Compaq installation CD following the iPAQ should contain Active Sync)
- Terminal emulator software (e.g. HyperTerminal 6.3 can be downloaded from *http://www.hilgraeve.com/htpe*)

The software you need on the iPAQ:

- CRL/OHH Boot loader 2.18.54
  (Downloadable from *http://familiar.handhelds.org/releases/v0.6/install/bootldr-2.18.54.bin*)
- Boot Blaster 1.18
  (Downloadable from *ftp://ftp.handhelds.org/pub/linux/compaq/ipaq/v0.30/BootBlaster_1.18.exe*)
- Linux Familiar 0.6 X/GPE Bootstrap Image
  (Downloadable from
  http://familiar.handhelds.org/releases/v0.6/install/bootgpe.jffs2)

We selected the X/GPE bootstrap image, but there are other alternatives. E.g. the OPIE bootstrap image is more visual appealing, but lacks a default terminal application (which has to be installed manually afterwards). There is also a simple bootstrap root image which contains only the basic elements of the Familiar distribution.

A.1.2 INSTALLATION PROCEDURE

We present here an easy-to-follow step-by-step guide to install Linux Familiar.

Step 1: Use the ActiveSync application on the PC to transfer the `BootBlaster_1.18.exe` and `bootldr-2.18.54.bin` to the default folder `My Pocket PC/My Documents` on the iPAQ. You may use the USB cable/cradle for a faster transmission.

Step 2: On the iPAQ, start the ActiveSync application. Select `TOOLS > OPTIONS` and change "`USB`" to "`115200 default`". Press "`OK`".

Step 3: Connect the iPAQ to the PC using the serial cable/cradle. Start the Boot Blaster application on the iPAQ that you transferred in step 1. Select `FLASH > SAVE BOOTLR` to make a safety backup of the currently installed boot loader. Then select `FLASH > SAVE WINCE.GZ` to make a safety backup of important data used by Windows CE. Then copy both the saved_bootldr.bin file and the wince.gz file from the iPAQ folder `My Pocket PC/My Documents` to the PC for safe keeping.

Step 4: Still using the Boot Blaster application on the iPAQ, select `FLASH > PROGRAM` and then select the boot loader image you transferred to the iPAQ in step 1. A new boot loader is now being installed. If the installation was not verified, select `FLASH > VERIFY` and make sure it doesn't display any error messages.

Step 5: Undock the iPAQ. Press and hold the center of the joy pad (the big button) while resetting the iPAQ (use the stylus pen or a similar object to press the reset button on the lower right of the iPAQ). The CRL Boot Loader menu will now appear on the display. Dock the iPAQ.

Step 6: Start a terminal emulator (e.g. the HyperTerminal 6.3) and establish a new connection over the COM1 port (assuming you connected the serial cable to the COM1 on the PC). Use the following settings on this connection: `Bps=115200, Databits=8, Parity=None, Stopbits=1 and FlowControl=None`. Initiate the connection.

Step 7: On the CRL Boot Loader on the iPAQ, press "*Boot Serial*". You should then see the text `boot>` on the terminal window on the PC. If not, press `ENTER` on your keyboard in the terminal window.

Step 8: In the terminal window, enter: `load root`

Step 9: The iPAQ will now await a file transfer. In the HyperTerminal application on the PC select `TRANSFER > SEND FILE` and select the bootstrap image (jffs2 file) you downloaded (that should be the Linux Familiar 0.6 X/GPE Bootstrap Image, according to the requirements we specified). Then select Xmodem as the protocol.

Step 10: Check that the installation procedure goes through the following three phases: ERASE, WRITE and VERIFY (and in this order).

Step 11: When the "`boot>`" prompt reappear, enter: `boot`

Step 12: Linux Familiar 0.6 with the GPE Palmtop Environment has now been installed and booted, and a screen calibration window should appear. Press the screen using the stylus pen at the designated locations. Next a login window will appear. Use the default login parameters: `user = root, password = rootme`.

Note that a post installation script has to be executed, but this requires that the iPAQ has established an Internet connection.

A.2 GETTING THE IPAQ ONLINE

Getting Internet connectivity on the iPAQ is essential, so two different methods of achieving this is presented in sections A.2.1 and A.2.2.

A.2.1 USING A PCMCIA WLAN CARD

The easiest way of getting Internet connectivity on the iPAQ is by using a PCMCIA WLAN card together with an access point (AP). We tried cards from several different vendors; 3Com, Lucent, Cisco and Zyxel.

The cards tested:

- 3Com 11 Mbps Wireless LAN OfficeConnect PC Card (3CRSHPW196)
- Lucent ORiNOCO PC Card Silver
- Cisco Aironet 350 Series WLAN PC Card (AIR-PCM350)
- Zyxel ZyAIR B-100 WLAN PC Card

The only cards we were able to install properly on the iPAQ was the once from Lucent and Cisco, as the driver for these cards are included in the Familiar distribution.

The PCMCIA modules used by Familiar 0.6 are: pcmcia-cs version 3.1.29-29 and pcmcia-modules version 2.4.18. These versions support "*hot swapping*" of PCMCIA cards, i.e. the Card Manager will automatically restart and load the proper drivers (if existent) when a card is inserted into the slot.

Configuring the WLAN settings are done by editing the `/etc/pcmcia/wireless.opts` file. We made two separate configurations for the WLAN, and stored these in the files `wireless.opts.fcareadhoc` and `wireless.opts.fieldcare`.

Two networks modes were defined:

- fieldcare : A managed wireless network using one access point connected to the university LAN.
- fcareadhoc : An ad-hoc wireless network.

The file `/etc/pcmcia/wireless.opts.fcareadhoc` contains the following:

```
case "$ADDRESS" in

*,*,*,*)
      ESSID="fcareadhoc"
      MODE="Ad-Hoc"
      ;;
```

The file `/etc/pcmcia/wireless.opts.fieldcare` contains the following:

```
case "$ADDRESS" in

*,*,*,*)
      ESSID="fieldcare"
      MODE="Managed"
      ;;
```

Then we constructed two scripts managing the initialization of these modes.

The `/bin/adhoc` script manages the ad-hoc mode:

```
#!/bin/sh
#
# Enables AD-HOC network mode
#
echo "Initiating Ad-Hoc mode:"
cp /etc/pcmcia/wireless.opts.fcareadhoc /etc/pcmcia/wireless.opts
echo "[!] Please wait while restarting network interface"
/etc/init.d/pcmcia restart
sleep 5
ifconfig eth0 192.168.1.1 netmask 255.255.255.0
```

While the `/bin/managed` script manages the managed network mode (i.e. connection with an AP):

```
#!/bin/sh
#
# Enables MANAGED network mode
#
echo "Initiating connection with access point"
cp /etc/pcmcia/wireless.opts.fieldcare /etc/pcmcia/wireless.opts
/etc/init.d/pcmcia restart
```

When executed, these scripts will change the network configuration between ad-hoc and managed. To check if the new configuration was properly set, issue the following commands:

```
# ipconfig eth0 (assuming your WLAN interface is eth0)
```

and

```
# iwconfig eth0 (assuming your WLAN interface is eth0)
```

The first command will display the IP address associated with this interface. If no IP address is associated, you may set an address manually using the `ipconfig` command.

```
# ifconfig eth0 192.168.1.1 netmask 255.255.255.0
```

Please note that in managed network mode, the access point is usually connected to a network which is using a DHCP server. This server will automatically issue an IP address to the requesting device. So it is not recommended to manually set an IP address in managed mode, since you will need a globally valid IP address to be able to access Internet resources. In our ad-hoc network we use internal addresses which will not be forwarded by routers (e.g. `192.168.*.*`).

The second command above (`iwconfig`) will display the wireless settings of the specified network interface.

In ad-hoc mode it is important that all the network interfaces that together create the ad-hoc network use the exact same Extended Service Set Identifier (ESSID), in our case that is "*fcareadhoc*". It is possible to manually set the ESSID by using the `iwconfig` command.

```
# iwconfig eth0 essid fcareadhoc
```

Test the connectivity by pinging some remote host.

```
# ping www.cnn.com
```

or in ad-hoc mode try pinging another host on the ad-hoc network.

A.2.2 USING TCP/IP OVER USB

In lack of a PCMCIA WLAN card, it is possible to get Internet connectivity through the USB cable/cradle included in iPAQ 3660 package.

Firstly you'll need a network interface driver for the USB cradle. A driver for Windows XP has been developed by Bahia Europe and is freely available (for personal use) at: *http://www.bahia21.com/shared/zip/bnd_0_9_9_1.zip*

Installation procedure:

Step 1: Unzip the Bahia driver to a directory of your choice.

Step 2: Undock the iPAQ. Open a terminal window on the iPAQ and load the USB-eth module:

```
# modprobe usb-eth
```

Step 3: Dock the iPAQ. Windows XP will detect a new unit on the system of the type "*Other Device*". When Windows asks for a driver, select the directory from step 1. If the installation of the device was successful, a secondary network interface (in our case called "*Local Area Connection 2*") has been added.

Step 4: In the terminal window on the iPAQ execute the following command:

```
# ifconfig usbf 192.168.0.2 netmask 255.255.255.0
```

This IP address will now be associated with the USB interface on the iPAQ.

Step 5: Then we need to specify a default gateway in the iPAQ's routing table.

```
# route add default gw 192.168.0.1
```

This IP address is the address of the Bahia network interface on the PC.

Step 6: We also need to specify a default name server on the iPAQ. We used the IP address of our university's primary DNS server.

```
# echo "nameserver 129.241.42.2" > /etc/resolv.conf
```

Step 7: On the PC select `START > CONTROL PANEL > NETWORK CONNECTIONS` and right click on the network interface connected to the Internet (often called "Local Area Connection 1"). Select `PROPERTIES > ADVANCED` and enable the checkbox "*Allow other network users to connect through this computer's Internet connection*".

Step 8: Verify that the IP address of the Bahia network interface on the PC is the same as the address used for the default gateway in step 5.

Step 9: In the terminal windows on the iPAQ, test the connectivity by pinging some remote host.

```
# ping www.cnn.com
```

When undocking the iPAQ it is recommend that you use Windows XP's Safely Remove Hardware function, otherwise you may encounter some problems when trying to re-dock.

The configuration is not entirely stable, so you may have to reconfigure the iPAQ when docking. We recommend doing the following procedure.

When the iPAQ is undocked executed these command to remove previous configurations:

```
# ifconfig usbf down
# rmmod usb-eth
```

Reload the module:

```
# modprobe usb-eth
```

and then dock the iPAQ in the cradle and reconfigure the network parameters

```
# ifconfig usbf 192.168.0.2 netmask 255.255.255.0
# route add default gw 192.168.0.1
```

Since Linux Familiar on the iPAQ is running an SSH daemon, it is just plain stupid (and annoying) to use the terminal application on the iPAQ and typing using the stylus pen with the virtual keyboard. Instead, a much better solution when operating the iPAQ is to use a SSH client on the PC that connects to the SSH daemon on the iPAQ.

When using a PCMCIA WLAN card in a managed network mode, all you have to do is enter the IP address of the iPAQ in your SSH client software on the PC to access it. When using TCP/IP over the USB connection the problem is that the iPAQ is using an internal IP address (`192.168.0.2`) and is therefore not accessible from the outside. Even though the PC is acting as a proxy for the iPAQ, it does not do traffic forwarding (yet). Since the PC is the iPAQ's gateway, all the forwarding configurations should be done on the PC. The procedure is as follows:

On the PC select: `START > CONTROL PANEL > NETWORK CONNECTIONS` and right click on the network interface connected to the Internet (often called "*Local Area Connection 1*"). Then select `PROPERTIES > ADVANCED > SETTINGS > ADD` and input the following data:

```
Description of service: SSH
Name/IP of computer hosting service: 192.168.0.2
External port: 22
Internal port: 22
Protocol: TCP
```

Now it should be possible to use SSH from the PC to access the iPAQ.

A.3 LINUX FAMILIAR POST INSTALLATION

Now that the iPAQ has Internet connectivity it's time to perform the post installation procedure. Open a terminal window on the iPAQ or use a SSH client from the PC and enter:

```
# ipkg update
```

This will update the package list on the iPAQ. A lot of information will be displayed, but with the current font type it is quite hard to read any of it, but not to worry the fonts will be fixed later. If desired, all installed packaged can now be updated to the newest version by executing:

```
# ipkg upgrade
```

Then it's time to run the post installation script, which will among other things correct the font type.

```
# /root/postinst
```

To enable all changes, either logout and login or reboot the iPAQ.

A.4 RESTORING WINDOWS CE ON A LINUX IPAQ

If you for some reason would like to restore the iPAQ back to the state it was in before installing Linux Familiar, i.e. running Windows CE, the procedure is as follows.

Step 1: Undock the iPAQ. Press the center of the joy pad while rebooting the iPAQ. The Boot Loader menu will now appear.

Step 2: Dock the iPAQ in the serial cradle and open a terminal window on the PC (e.g. Hyper Terminal). Press "*Boot serial*" on the iPAQ to establish a connection to the PC. Press enter on the keyboard if the boot prompt does not appear in the terminal window.

Step 3: `#boot> load boot`

Step 4: Then start an Xmodem file transfer from the terminal window on the PC and transfer the `wince_image.gz` backup file which was created by Boot Blaster before installing Linux.

Step 5: The rest should now proceed automatically. When WinCE is restored and rebooted you may use the "*Self Test*" functionality in WinCE to verify the contents in the Flash memory.

A.5 JAVA ON IPAQ

The iPAQ H3660 is using an Intel StrongARM SA-1110 RISC processor [Com00] and currently Sun Microsystems do not provide a *Java Runtime Environment* (JRE) for this architecture. However The Blackdown Project [VDJ$^+$03] has ported Sun's JRE 1.3.1 to the StrongARM architecture, with the Familiar Linux on an iPAQ specifically in mind.

A.5.1 SOFTWARE NEEDED

- Blackdown JRE 1.3.1 RC1
  (Downloadable from: *http://www.mirror.ac.uk/sites/ftp.blackdown.org/java-linux/JDK-1.3.1/arm/rc1/j2re-1.3.1-RC1-linux-arm.tar.bz2*)
- Blackdown Additional iPAQ Stuff
  (Downloadable from: *http://www.mirror.ac.uk/sites/ftp.blackdown.org/java-linux/JDK-1.3.1/arm/rc1/additional-ipaq-stuff.tar.gz*)
- X Toolkit Library for Linux Familiar
  (Downloadable from: *http://handhelds.org/download/packages/libxt/libxt_4.2-2_arm.ipk*)
- GNU Standard C++ Library for Linux Familiar
  (Downloadable from: *http://handhelds.org/download/packages/libstdc++2.10-glibc2.2/libstdc++2.10-glibc2.2_2.95.4-0.010407_arm.ipk*)
- RXTX binaries
  (Downloadable from: *http://www.linux.org.uk/~taj/rxtx-bins.1.tar.gz*)
- Sun Solaris version of Java COMM API
  (Downloadable from: *http://java.sun.com/products/javacomm/*)

A.5.2 INSTALLATION PROCEDURE

Since the JRE tar file (`j2re-1.3.1-RC1-linux-arm.tar.bz2`) is about 12 MB, it is probably best to unpack it on your PC and then transfer the files to the iPAQ using e.g. the `SCP` command on the iPAQ (part of the openSSL package) or the WinSCP application [Pri03].

```
# scp -r <user>@<ip address>:/dir/to/unpack/jre/. /destination/dir/
```

In our case the command looked like this:

```
# scp -r toringe@129.241.209.106:/home/toringe/jre131arm/. /mnt/ramfs/
usr/local/bin/jre131
```

Notice that the JRE is stored in the RAM of the iPAQ (`/mnt/ramfs`) and not in the non-volatile Flash ROM, due to the limited storage capacity of the ROM. Linux Familiar occupies about 11 MB of the ROM, and the iPAQ's Asset Information (static data) occupies 0.5 MB, leaving not enough space for the JRE. The consequence of this is that the JRE installation will be lost when reboot the iPAQ.

When the JRE is in place, we'll need to update the `PATH` and `CLASSPATH` variables.

```
# export PATH=/mnt/ramfs/usr/local/bin/jre131/bin:$PATH
# export CLASSPATH=.:/mnt/ramfs/usr/local/jre131/lib/i18n.jar:/mnt/ram
fs/usr/local/jre131/lib/rt.jar:/mnt/ramfs/usr/local/jre131/lib/sunrsas
ign.jar:$CLASSPATH
```

This update is only valid for the current login session. To make the update permanent, edit the `/etc/profiles` file and add a similar configuration there.

Then we'll need to install the "Additional iPAQ stuff". Unpack the `additional-ipaq-stuff.tar.gz` file in the root (`/`) directory. This is important as the packed file contains the proper directory structure needed for a successful install.

```
# cd /
# tar zxvf additional-ipaq-stuff.tar.gz
```

Run ldconfig to configure the runtime bindings, so that the JRE is able to locate the `libBrokenLocal` library.

```
# ldconfig
```

And finally we need to install the two extra libraries; `libXt` and `libstdc++`.

```
# ipkg install libxt_4.2-2_arm.ipk
# ipkg install libstdc++2.10-glibc2.2_2.95.4-0.010407_arm.ipk
```

If no errors appeared, JRE 1.3.1 should not be installed and working properly.

A.5.3 JAVA SECURE SOCKET EXTENSION (JSSE)

JSSE is included in JRE and SDK 1.4, so there shouldn't be any problems using SSL on the CMDA and FCC units, as these are standard PCs. But the iPAQ is using

Blackdown's ported JRE, which is based on Sun's JRE 1.3.1. And this version does not have JSSE included.

Sun has released a JSSE package for the Java 2 SDK version 1.2.x and 1.3.x [Sun03b], but this release is incompatible with the ported JRE version by Blackdown. Hopefully this will be correct in the upcoming versions from Blackdown.

We have added an element to the `config` file of FieldCare, enabling and disabling the use of SSL.

```
[Security]
SSL = {on, off}
```

A.5.4 SUN JAVA COMM API

The FieldCare uses "Java Communications API" for communication with the iButton. When trying to run the program on Linux, we experienced some problems. These were all solved when we found out that we couldn't use SUN's standard version of the API on Linux. We found a very helpful guide [Was02], and to get the communication up and running on Linux we need the following:

We first need to have JRE in place. This was explained above in A.5.

The next we have to do is install RXTX, which are open source serial and parallel I/O libraries supporting Sun's CommAPI [Jar03].

To install RXTX, we first unpack the binary distribution (`rxtx-bins.1.tar.gz`), as this contains a precompiled version specifically to the StrongARM architecture of the iPAQ. Since the iPAQ has limited storage capacity, it may be a good idea to unpack the files on your PC and then transfer what you need to the iPAQ using e.g. the `SCP` command on the iPAQ.

We need to copy the shared objects into our java installation:

```
# scp root@129.241.209.106:/unpacked/rxtx-bins.1/1.4/armv4l-unknown-
linux-gnu/libSerial-1.4.15.so /mnt/ramfs/usr/local/jre131/lib/armv41/
```

The next we need to do is to install the jcl.jar file:

```
# scp root@129.241.209.106:/unpacked/rxtx-bins.1/1.4/jcl.jar
/mnt/ramfs/usr/local/jre131/lib/ext/
```

The installation of RXTX is then complete.

We then need to install Java Comm API. It is important that we selected the Solaris version of this API. As before we unpack the file and transfer what we need from the computer to the iPAQ:

```
# scp root@129.241.209.106:/unpacked/commapi/comm.jar
/mnt/ramfs/usr/local/jre131/lib/ext/
```

The last thing we have to do is create the properties file that the Comm API will use to load the drivers (.so files)

```
# echo "Driver=gnu.io.RXTXCommDriver" >
/mnt/ramfs/usr/local/jre131/lib/javax.comm.properties
```

The Java Comm is now installed.

A.6 PREPARING THE CMDA AND FCC DEVICES

In the FieldCare prototype this thesis has based its implementation on, the CMDA device is a tablet PC. Since we didn't have any tablet PC, we used a laptop PC (Toshiba Portégé 3440 CT) to represent the CMDA and for the FCC device we used a standard PC. These two devices had the following general specification:

- CMDA:
  Intel Pentium III Mobile 500 MHz CPU
  192 MB RAM
  30 GB Hard drive
  Cisco Aironet 350 Series PCMCIA 802.11b WLAN Card

- FCC:
  AMD Athlon XP 1700+ CPU
  256 MB RAM
  20 GB Hard drive
  Cisco Aironet 350 Series PCI 802.11b WLAN Card
  3Com EtherLink 10/100 PCI Network Card

The FCC had two network interface cards since this device acts as a gateway between the ad-hoc WLAN and the external network (e.g. Internet).

A.6.1 INSTALLING LINUX

Linux Red Hat 9.0 was installed on both the CMDA and FCC machine. A custom installation was selected, and we installed a minimal set of software. Among other things installed was the XFree86 Window System, Samba (for easy connectivity), OpenSSH, Wireless Tools and Pico (for editing). It is very important to correctly set the swap partition used by Linux. A rule of thumb is that the swap partition should be at least two times the amount of RAM on the system.

A.6.2 INSTALLING JAVA

We installed Java J2SE version 1.4.1_02 SDK, which was distributed by Sun as a RPM file. As the CMDA and FCC not necessarily need the complete SDK version, installing the JRE is recommended. The installation was straight forward as any other RPM installations.

```
# ./j2sdk-1_4_1_02-linux-i586-rpm.bin
# rpm -ivh j2sdk_1_4_1_02-fcs-linux-i586.rpm
```

Then update the `.bash_profile` file in your home directory to include the path to the java bin directory and also define a `CLASSPATH`.

To be able to use the iButton on the CMDA and FCC, we need to install RXTX. This is basically the same procedure as was presented in A.5.4, however we need to select the i386 binaries of RXTX instead of the StrongARM compiled versions. The Solaris version of the Comm API is also needed for this installation.

A.6.3 INSTALLING THE CISCO DRIVER AND UTILITY

We downloaded the driver software and utility from Cisco's homepage by using the Software Selector at http://www.cisco.com/pcgi-bin/Software/WLAN/wlplanner.cgi. The driver for both the PCMCIA and PCI card is distributed as one single packed file called Linux-ACU-Driver-v2.0.tar.gz. According to the manual [CS02], we need the kernel-source and gcc to be able to properly install the driver, hence we download and install these first. We need to configure the up2date application to be able to download the kernel-source as it automatically skips all kernel related packages.

```
# up2date --configure
> 6
> C
> <ENTER>
#up2date kernel-source
#up2date gcc
```

Then we unpack the Cisco driver and utility file and execute an included script to compile and install it.

```
# tar zxvf Linux-ACU-Driver-v2.0.tar.gz
```

There are two scripts included; `kpciinstall` and `cwinstall`. The former script is for the PCI clients and the latter one for PCMCIA clients. We will here show the installation of the PCI client (i.e. the WLAN card on the FCC) as the installation is almost exactly the same for a PCMCIA client (i.e. the WLAN card on the CMDA) except for the name of the install script.

```
# sh kpciinstall
```

After the driver has compiled and the utilities have been installed, type the following to start the driver.

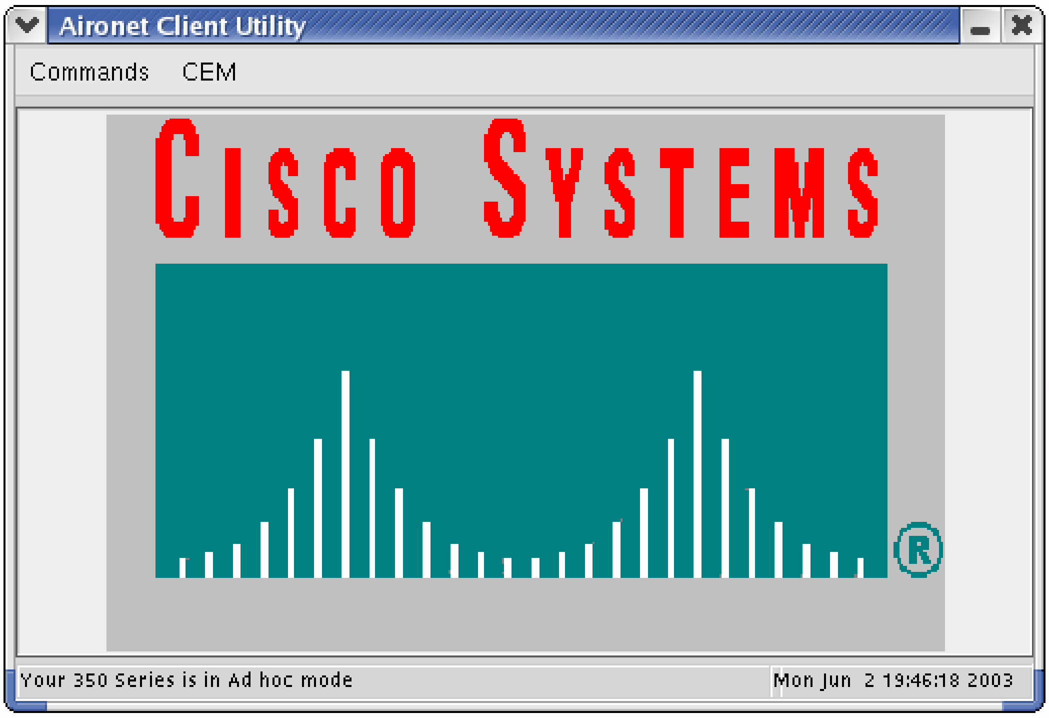
`# insmod airo` (use airo_cs instead for a PCMCIA client)

Configure the wireless interface by using the ipconfig command

```
# ipconfig 192.168.1.250 netmask 255.255.255.0
```

Then we launch the ACU (see Figure A.1 below) to configure the WLAN card. We use the `&` to fork the application to a separate process.

```
# /opt/cisco/bin/acu &
```



Your 350 Series is in Ad hoc mode

Mon Jun 2 19:46:18 2003

Figure A.1 – The Cisco ACU application

Select `COMMANDS > EDIT PROPERTIES` and then set the `Network Type` to `Ad Hoc`. Also specify the `Client Name` and `SSID1` parameters (remember that all nodes in the ad-hoc network must use the same `SSID` or `ESSID`).

Then reboot the machine. Please note that after each reboot you have to start the driver, configure the wireless interface and start the ACU manually.

A.7 ENABLING WEP IN LINUX

By using the `iwconfig` tool in Linux, it is easy to enable WEP encryption. Both 40 bit keys and 104 bit keys are supported (assumed the WLAN card support it), and when adding the 24 bit initialization vector, the result is a 64 bit and 128 bit encryption respectively.

The WEP key can either be specified as a hexadecimal number or as a string. Enter a hexadecimal of length ten or a string with five characters for 40-bit key or enter a hexadecimal of length twenty-six or a string with thirteen characters for 104-bit key.

Examples of all four possibilities (remember to change to more appropriate key values):

```
#iwconfig eth1 key 0123-4567-89 restricted
#iwconfig eth1 key s:passw restricted
```

```
#iwconfig eth1 key 1234-1234-5678-5678-9090-9090 restricted
#iwconfig eth1 key s:mypassphrases restricted
```

The above commands will se the specified key as the new current key, and also enable encryption if it was previously disabled. The `restricted` mode increases security by disregarding all non-encrypted packets.

When a key has been specified you may use the parameters on and off to respectively enable and disable encryption.

```
# iwconfig eth1 key on
# iwconfig eth1 key off
```

Remember to set the restricted encryption mode if the mode was by default set to open.

```
# iwconfig eth1 key restricted
```

To enable WEP on the CMDA and FCC machine, you have to use the Cisco ACU application.

A.8 INSTALLING THE OLSR DAEMON

The OLSR implementation we installed on all WLAN devices was developed by the U.S. Naval Research Laboratory (NRL) [WDM$^+$03]. The code base of this implementation was based on the Version 3 of INRIA's Optimized Link State Routing protocol draft for the IETF. NRL has among other things precompiled the OLSR daemon for Linux Red Hat as a RPM file, and for Linux Familiar running on a StrongARM processor. So all we had to do was to install the binaries on the devices. All files were downloaded from http://pf.itd.nrl.navy.mil/projects/olsr/, and we used the 1.0a9 version of the implementation.

A.8.1 OLSR ON THE MDA

Since the version available for the StrongARM processor already had been compiled into a binary file, all we had to do was to transfer this file to the iPAQ. This can be done in various ways. Either use the Hyper Terminal for a Xmodem transfer over the serial cable, or use the `scp` command or WinSCP to copy the file over an SSH connection. The example below uses the scp command.

```
# scp armolsrd root@192.168.1.1:/usr/sbin/
```

To execute the OLSR daemon on the iPAQ, use the following command.

```
# /usr/sbin/armolsrd -d 2 -i eth0
```

The `-d 2` mode represents debug level 2. Execute it without the `-d` option to fork it to the background as a separate process. We created two simple scripts to start and stop the OLSR daemon called `olsrd` and `killolsrd`. These two scripts are also used by the FieldCare application, and they are described in Appendix B.

A.8.2 OLSR ON THE CMDA AND FCC

The precompiled daemon to Linux Red Hat was distributed as a RPM file, so the installation was quite easy, and it had no prerequisites.

```
# rpm -ivh nolsrd-1.0a9-1.i386.rpm
```

The daemon binary was installed in `/usr/sbin/` as `nolsrd`. To execute this OLSR daemon the following command may be used (assuming `eth1` is the wireless interface)

```
# /usr/sbin/nolsrd -d 2 -i eth1
```

Again the `-d` option specifies the debug level and without using this option, OLSR will fork to the background. We created two simple scripts to start and stop the OLSR daemon called `olsrd` and `killolsrd`. These two scripts are also used by the FieldCare application, and they are described in Appendix C.

A.9 PROBLEM WITH SERIAL PORT ON LINUX FAMILIAR

We realized a problem with the serial port (`dev/ttySA0`) on the iPAQs running Linux Familiar 0.6, when we tried to connect to an iButton on the Blue Dot receptor through a PCMCIA serial adaptor. This problem results in the fact that we are then unable to use an iButton to login on the iPAQs.

The problem was initially believed to be related to the 1-Wire API used by the application trying to access the iButton, however it was later realized that the problem was probably rather related to Linux Familiar's serial port drivers.

The following entry existed in the `/etc/pcmcia/config` file:

```
device “serial_cs”
     class “serial” module “serial_cs”
```

So, the client driver file `serial_cs` had been defined in the PCMCIA config file. However when we tried to insert and reinsert the PCMCIA serial adaptor card, the following entries was made in the `/mnt/ramfs/var/run/stab` file.

Before we inserted the PCMCIA serial adapter card in the second PCMCIA socket of the iPAQ, the file contained this:

```
Socket 0:   Cisco Aironet 350
0           network airo_cs 0 eth0
Socket 1:   empty
```

Then we inserted the serial adapter card:

```
Socket 0:   Cisco Aironet 350
0           network airo_cs 0 eth0
Socket 1:   Serial or Modem
```

So, we see here that the PCMCIA serial adapter card has been detected by cardctl. If we did the same thing on a laptop running Ret Hat 9.0, we had the following entries in the `/var/lib/pcmcia/stab` file before we inserted the card.

```
Socket 0:   350 Series WLAN
0           network airo_cs 0 eth0
Socket 1:   empty
```

When we inserted the card:

```
Socket 0:   350 Series WLAN
0           network airo_cs 0 eth0
Socket 1:   Serial or Modem
1           serial serial_cs        0     ttyS2 4     66
```

We were unable (due to lack of time) to investigate this problem any further. However we had to implement a hack to temporarily correct this problem. So we included another run mode of the FieldCare application, namely the `-B` mode for “Bypass”.

This mode will bypass the iButton login sequence and instead read the user information for a static file instead (called `login.data`). This file contains all the info from an iButton, including a hash value of the PIN code, instead of the code itself. To easy the modification of this `login.data` file, a custom PIN code may be generated by using the `HashPin` application. This program will take a PIN code as an input and provide the SHA-1 hash value as the output. In Linux, use the `HashPinExecutor` script to securely input the PIN code into the shell. But all this should not be considered a solution to the problem; it’s just a simple hack for demonstration purposes.

This problem may be solved in later versions of the Linux Familiar; otherwise a suggestion would be to investigate if the kernel compiled on the device has an integrated support for the serial driver. The serial client driver `serial_cs` requires the kernel serial driver to be enabled with `CONFIG_SERIAL`. This driver may be built as a module.

A.10 FIELDCARE APPLICATION USER MANUAL

The new FieldCare prototype is dependent on Linux[13], and it now consists of three separate external modules. It is important how these modules are started. This section will explain how to get FieldCare up and running.

A.10.1 STARTUP PROCEDURE

To be able to run FieldCare properly you need to follow the procedure presented here.

1. *Start the RMI Registry*

   The `rmiregistry` program should be started in the same directory as InfoServer. This is done in order for the registry to properly load the stub files.

[13] FieldCare is still able to run on Windows, but not in dynamic mode (i.e. using OLSR).

Alternatively, the absolute path of the stub files could be entered into the `CLASSPATH` variable.

NB! On some Linux installations (e.g. Red Hat 9), there will exist an auxiliary `rmiregistry` program located in `/usr/bin/rmiregistry`, which is not Sun's `rmiregistry`. So when starting the `rmiregistry` on the CMDA and FCC, always used the absolute path of the rmiregistry in your `%JAVAHOME` directory.

```
# /usr/java/j2sdk1.4.1_02/jre/bin/rmiregistry.
```

2. *Start the InfoServer*

The easiest way to start the InfoServer is by using the run files located in the same directory as the InfoServer. If you decide to run it manually, always use the following parameters.

```
# java -Djava.security.policy=InfoServerSecurityPolicyNoSSL
-Djava.rmi.server.hostname= 192.168.1.2 Server
```

Since the InfoServer implements the RMI Security Manager, a security policy file must be used. The example above uses the "no SSL" policy, because of the JSSE problem described in 6.6.4. The `hostname` property is used to properly bind the correct network interface in the rmiregistry (we experienced some communication problems when not using this property).

3. *Start the External InfoServer* (Only applicable to the FCC).

It is necessary to run the External InfoServer on the FCC device, as this device has two network interfaces (one to the internal WLAN, and the other to the FCCDB). Because of this, the External InfoServer will be bound to the FCCDB interface, while FCC's InfoServer (from step 2.) will be bound to the WLAN interface. The easiest way to start the External InfoServer is by using the run files (e.g. `run.external`). If executed manually, the same properties as used in step 2 above should be applied. However, the `hostname` property and the class file name must be changed.

```
# java -Djava.security.policy=InfoServerSecurityPolicyNoSSL
-Djava.rmi.server.hostname= 192.168.2.1 ExternalServer
```

4. Start FieldCare

Now as the InfoServer is running and properly bound to the registry, the FieldCare application may be started.

```
# java -noverify -jar Fieldcare.jar <mode>
```

The `noverify` parameter was used to circumvent a problem with the RXTX library by disabling the bytecode verifier. Bytecode verification is performed by the Java interpreter to ensure the security, integrity, and correctness of the code that it executes. It consists of a series of tests that verify that the code can be safely executed. The current version of RXTX had an incompatibility problem with this verification sequence, but hopefully this will be corrected in a later version, as this problem is related to the general java security. The different modes FieldCare can be executed in are presented in the following section.

A.10.2 FIELDCARE START MODES

When starting FieldCare, there are different starting modes.

- *Dynamic Mode (Default)*

  The default mode for FieldCare is the dynamic mode (no parameters required). This mode will run FieldCare and use the OLSR routing protocol. Access control with iButtons is enabled.

- *Static Mode*

  Starting FieldCare with a `-S` will make it run in static mode. This is used when you want to run the program on a Windows computer. Using this mode will disable the use of dynamic routing since we haven't been able to get OLSR to work as planned (i.e. by updating the routing table) in Windows, and instead use the old static routing as configured in the config file. Access control with iButtons is enabled.

- *Bypass Mode*

  By starting FieldCare with a `-B` it will run in a mode bypassing the access control. It will use a login file (called login.data) containing the required user information, including the hash-value of the user's PIN code. This mode was implemented because of the problem with the serial port on the iPAQ as described in A.9.

Some device specific examples are:

MDA: `# java -noverify -jar FieldCare.jar -B`

CMDA/FCC: `# java -noverify -jar FieldCare.jar`

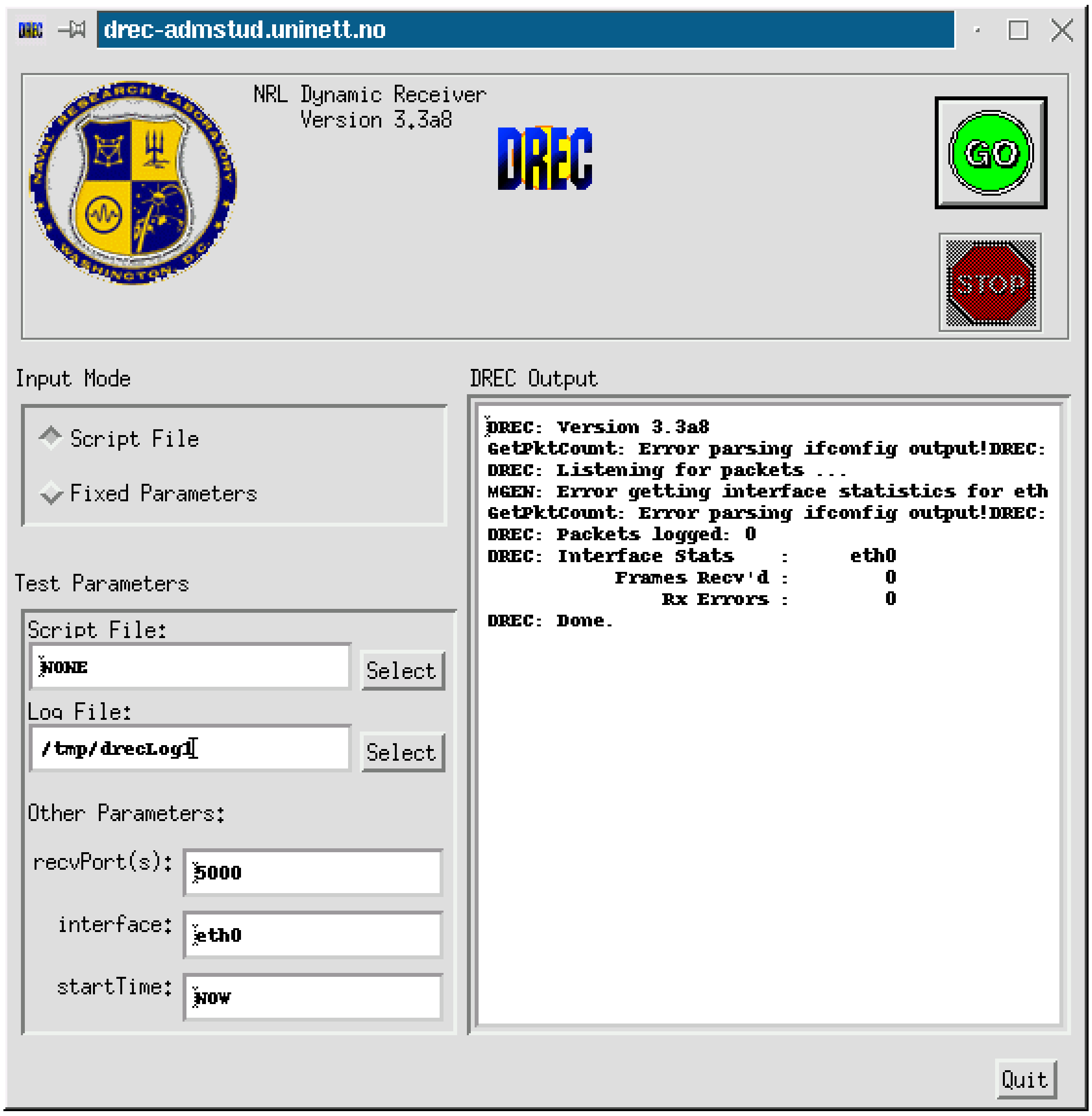
Now FieldCare should be up and running, notice that the network communication will not start until the user has been authenticated by using the PiTag (iButton) to log on.

APPENDIX B

This appendix presents the tools used in the OLSR test and also the results of the test.

B.1 THE MGEN / DREC TOOL

The Multi-Generator (MGEN) is a open source network tool developed by the Protocol Engineering Advanced Networking Research Group (PROTEAN) at the Naval Research Laboratory. MGEN provides the ability to perform IP network performance tests and measurements using UDP/IP traffic. The toolset generates real-time traffic patterns so that the network can be loaded in a variety of ways. The generated traffic can also be received and logged for analyses [AG03].



Script File: NONE

Log File: /tmp/drecLog1

Other Parameters:

recvPort(s): 5000

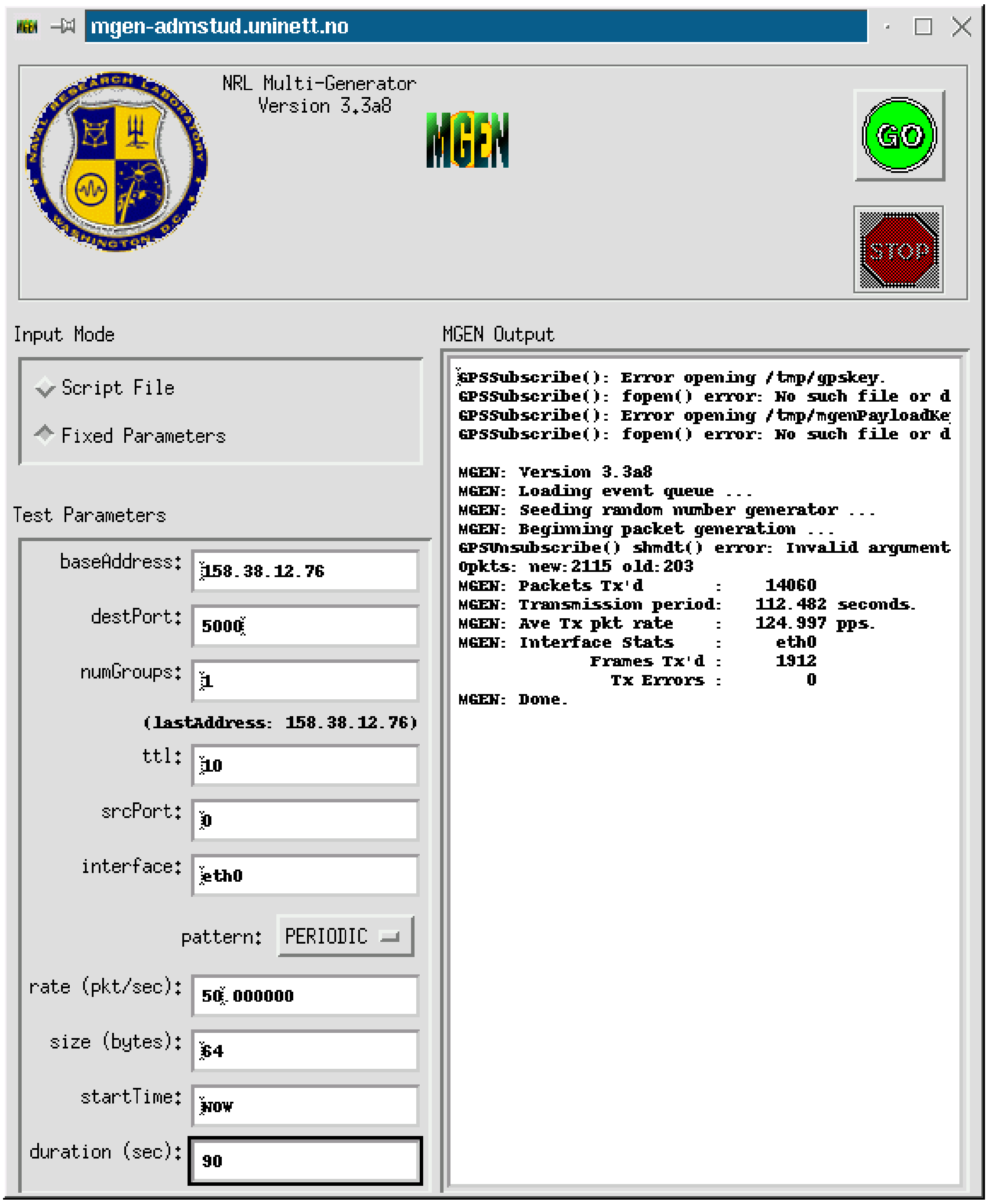
interface: eth0

startTime: NOW

DREC Output

```
DREC: Version 3.3a8
GetPktCount: Error parsing ifconfig output!DREC:
DREC: Listening for packets ...
MGEN: Error getting interface statistics for eth
GetPktCount: Error parsing ifconfig output!DREC:
DREC: Packets logged: 0
DREC: Interface Stats      :      eth0
           Frames Recv'd   :         0
               Rx Errors   :         0
DREC: Done.
```

Figure B.1 – Screenshot of the MGEN application



Input Mode

- Script File
- Fixed Parameters

Test Parameters

baseAddress:	158.38.12.76
destPort:	5000
numGroups:	1
	(lastAddress: 158.38.12.76)
ttl:	10
srcPort:	0
interface:	eth0
pattern:	PERIODIC
rate (pkt/sec):	50.000000
size (bytes):	64
startTime:	NOW
duration (sec):	90

MGEN Output

```
GPSSubscribe(): Error opening /tmp/gpskey.
GPSSubscribe(): fopen() error: No such file or d
GPSSubscribe(): Error opening /tmp/mgenPayloadKe
GPSSubscribe(): fopen() error: No such file or d

MGEN: Version 3.3a8
MGEN: Loading event queue ...
MGEN: Seeding random number generator ...
MGEN: Beginning packet generation ...
GPSUnsubscribe() shmdt() error: Invalid argument
Opkts: new:2115 old:203
MGEN: Packets Tx'd          :     14060
MGEN: Transmission period:    112.482 seconds.
MGEN: Ave Tx pkt rate       :   124.997 pps.
MGEN: Interface Stats       :      eth0
                 Frames Tx'd :      1912
                   Tx Errors :         0
MGEN: Done.
```

Figure B.2 – Screenshot of the DREC application.

We[14] installed the version 3.3 of the toolset with graphical interface. This toolset consists of two primary applications; *MGEN* and *DREC*. MGEN is the generator of real-time traffic patterns, while the DREC is the designated receiver of this traffic and generates traffic logs.

[14] The installation was actually done by a fellow student, Ronny Tafjord, who writes a master thesis on the performance issues of OLSR, and the testing was done in cooperation with him.

B.2 RESULTS FROM THE OLSR TEST

Please refer to section 5.4.2 for details on how the test was carried out. All graphs show the measured data rate over the time period of the transmission. The significant drops are the result of one relay node being taken offline, and the convergence time is measured as the time of ~0 data rate. Each configuration is run three times in order to compute an average value of the results.

B.2.1 USING DEFAULT PARAMETERS

The following three graphs shows the results when using the default protocol parameters:

```
hello interval = 0.500000
hello jitter = 0.100000
tc interval = 2.000000
tc jitter = 0.500000
polling interval = 0.050000
neighbor_hold_time = 4.000000
topology_hold_time = 10.000000
neighbor_timeout_mult = 8
topology_timeout_mult = 5
tos setting = 16
```

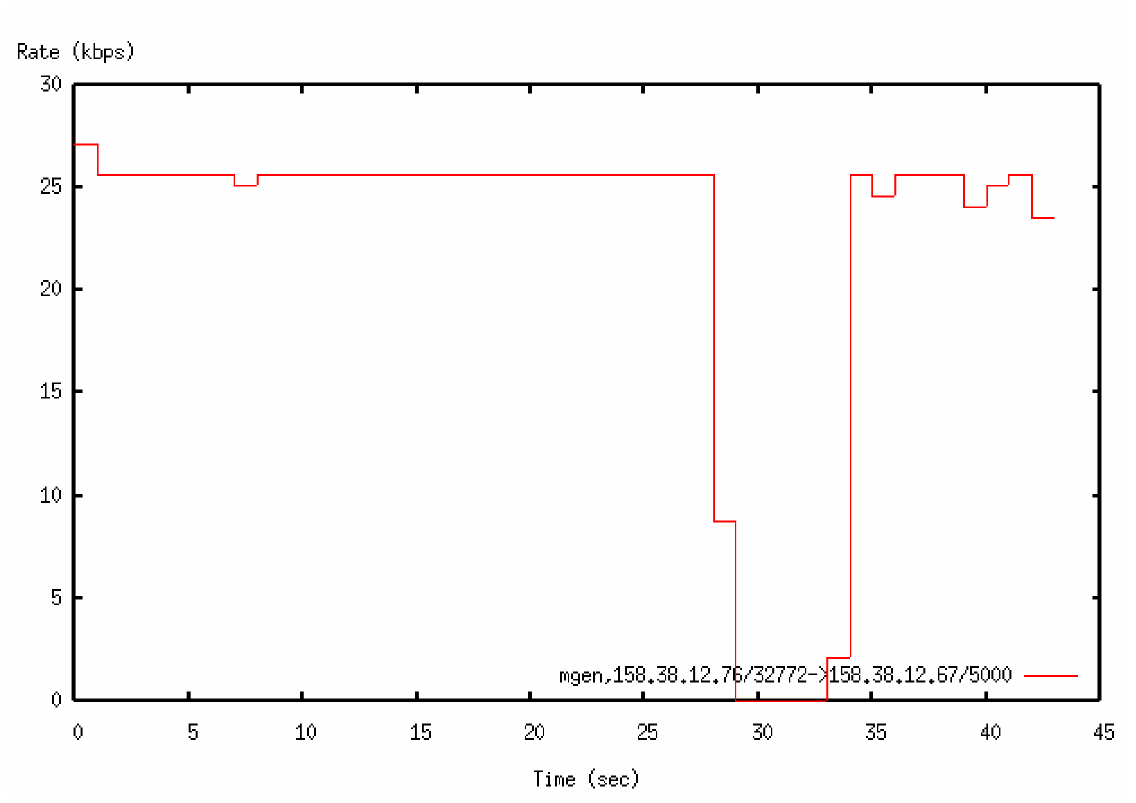


Figure B.3 – OLSR test run #1 using default parameters

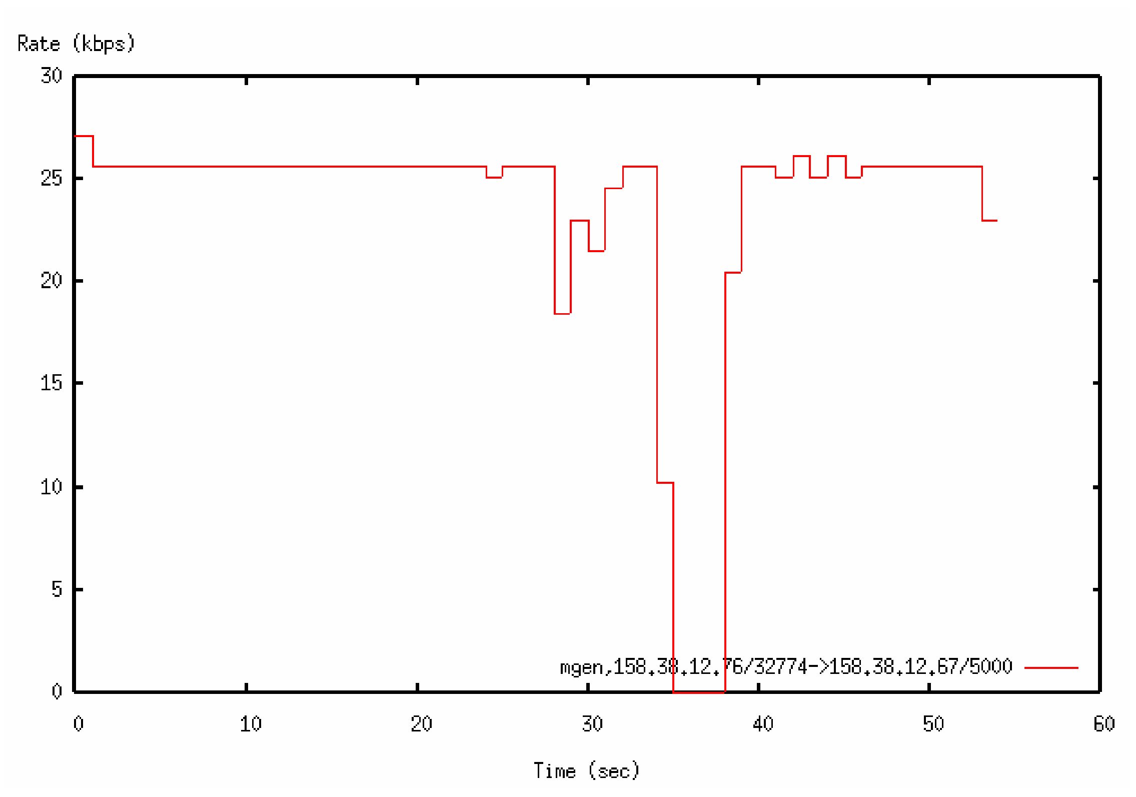


Figure B.4 – OLSR test run #2 using default parameters

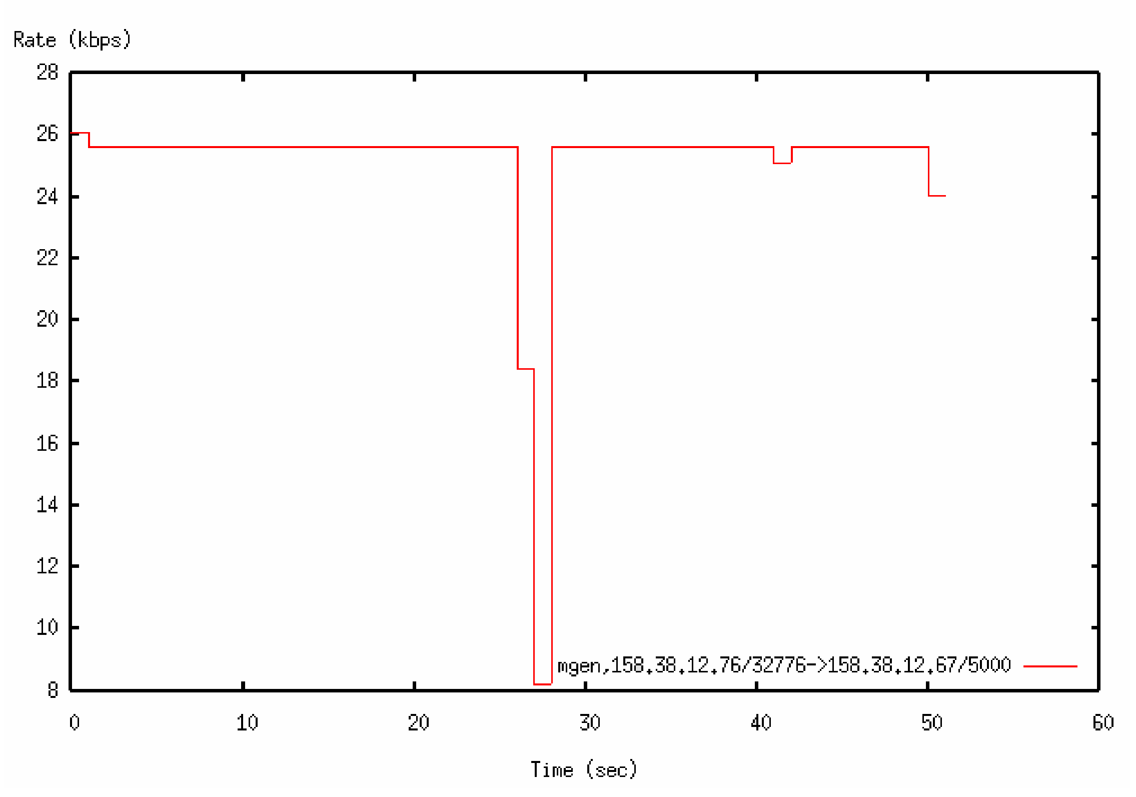


Figure B.5 – OLSR test run #3 using default parameters

The measured convergence time, when using the default protocol parameters, are shown in Table B.1.

	Convergence Time (s)
Test run #1	4
Test run #2	3,5
Test run #3	1
Average	2,8

Table B.1 – Average convergence time when using default parameters

B.2.2 USING CUSTOM PARAMETER SET

The following three graphs shows the results after we changed the protocol parameters (`-hint` and `-tcint`):

```
hello interval = 0.250000
tc interval = 1.000000
```

This increases the rate of updates done by OLSR (that is mainly done by the `tc interval` parameter), and should result in a smaller convergence time.

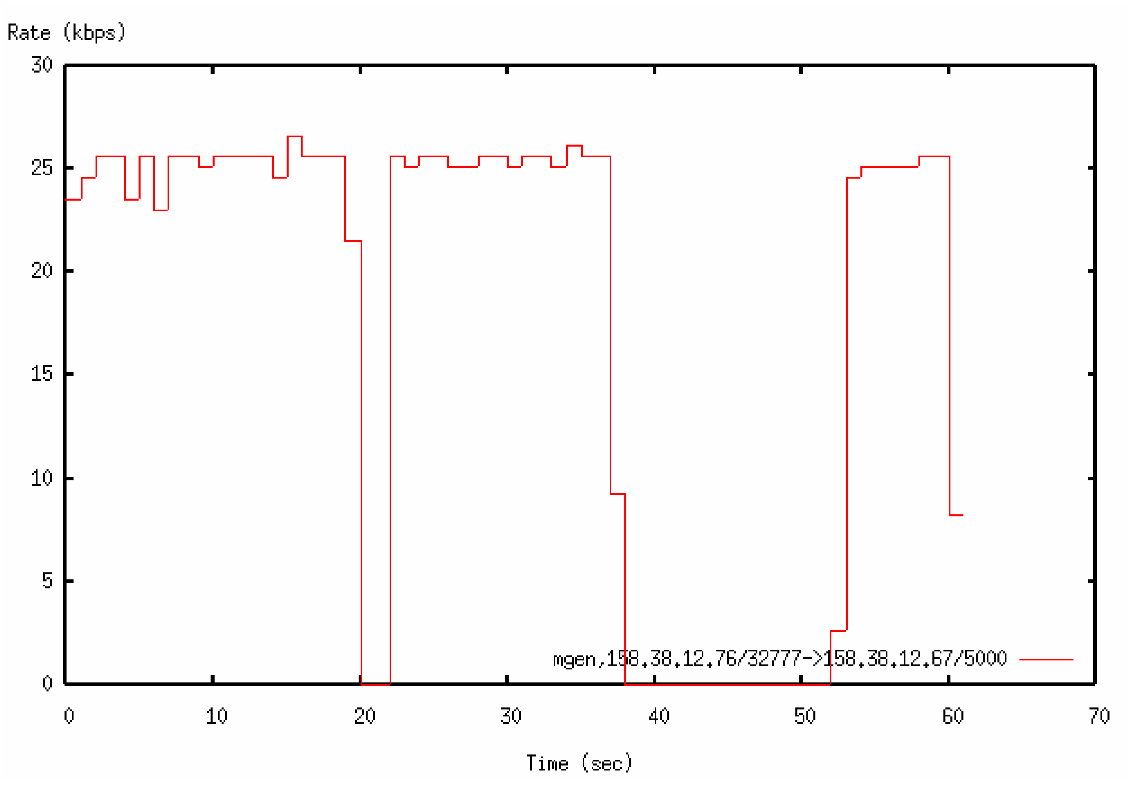


Figure B.6 – OLSR test run #1 using custom parameters

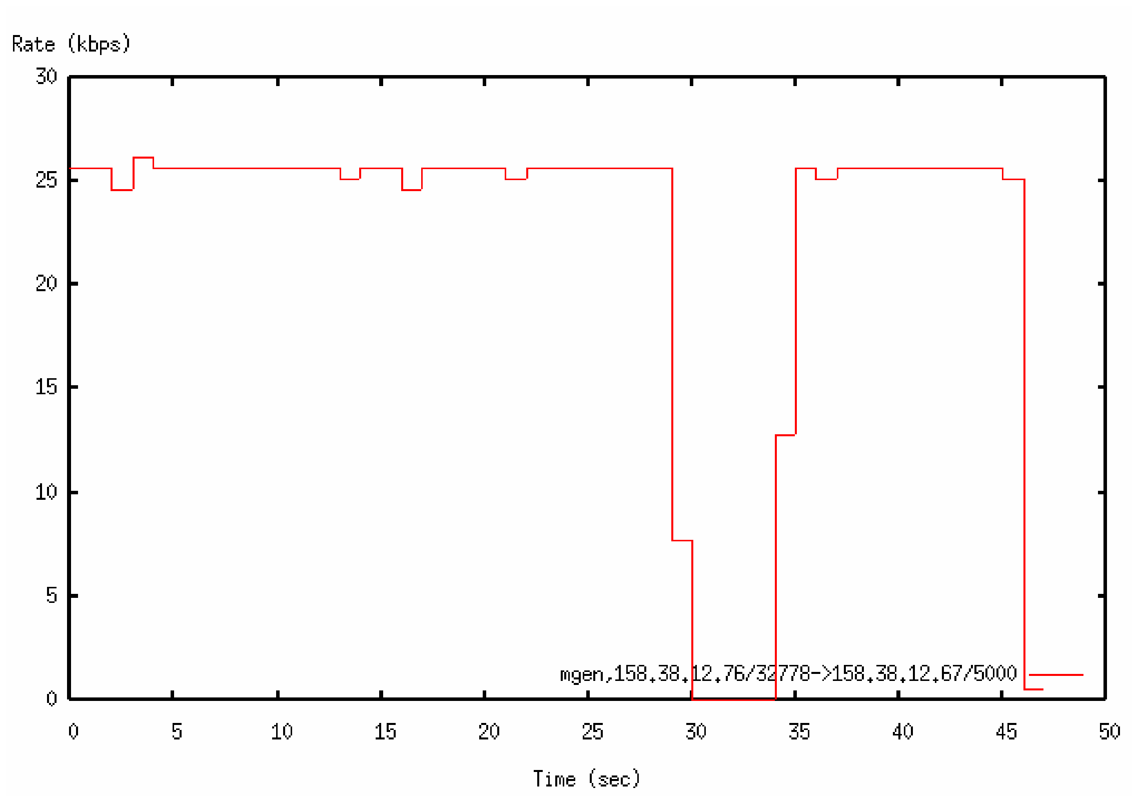


Figure B.7 – OLSR test run #2 using custom parameters

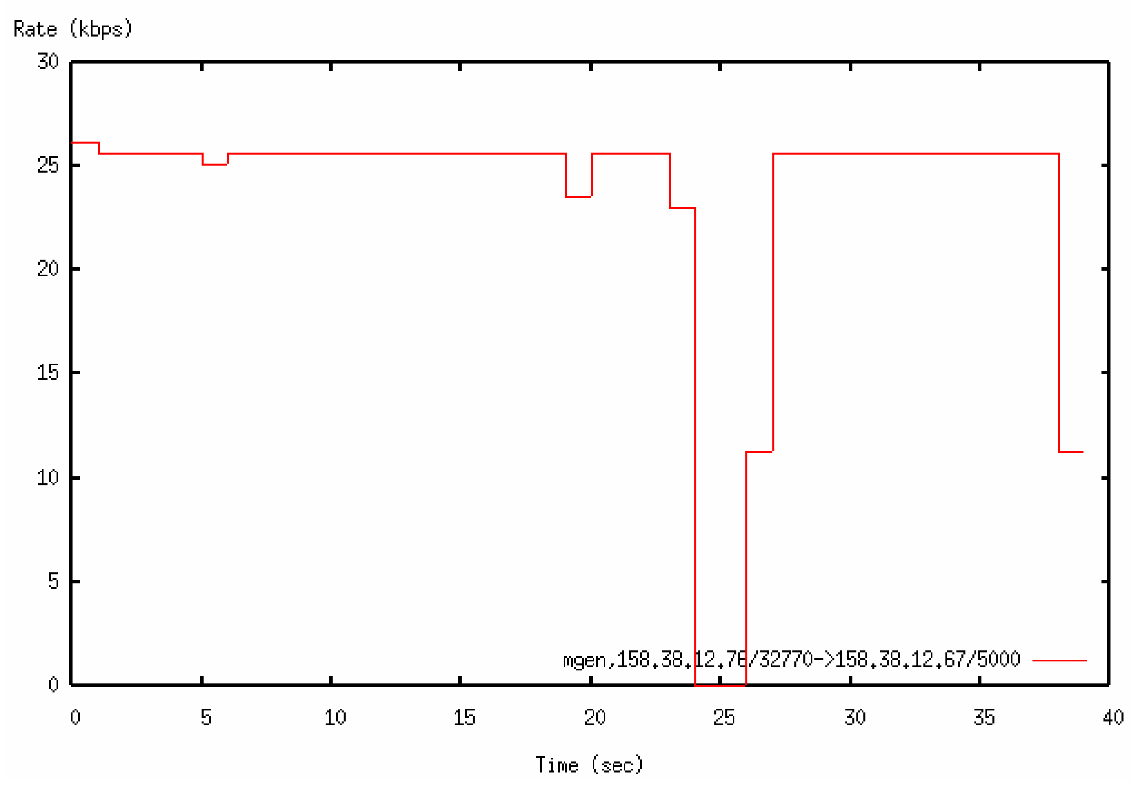


Figure B.8 – OLSR test run #3 using custom parameters

Notice the apparently strange results in Figure B.. This is probably due the amount of interference from the surrounding area of the test, and also due to some interoperability problems when using WLAN cards from different manufacturers. But the convergence time is still visible from the figures as it is always the first gap that drops all the way down to zero.

	Convergence Time (s)
Test run #1	2
Test run #2	4
Test run #3	2
Average	2,7

Table B.2 – Average convergence time when using modified parameters

As Table B.2 shows, the convergence time has not improved significantly by modifying the interval parameters, but had we done much more test runs (e.g. 15-20) we would probably have seen a greater improvement. The theory is that decreasing the tc interval will result in a more frequent transmission of Topology Control (TC) messages, and hence a more rapid adaptation to topology changes.

APPENDIX C

C.1 JAVADOC

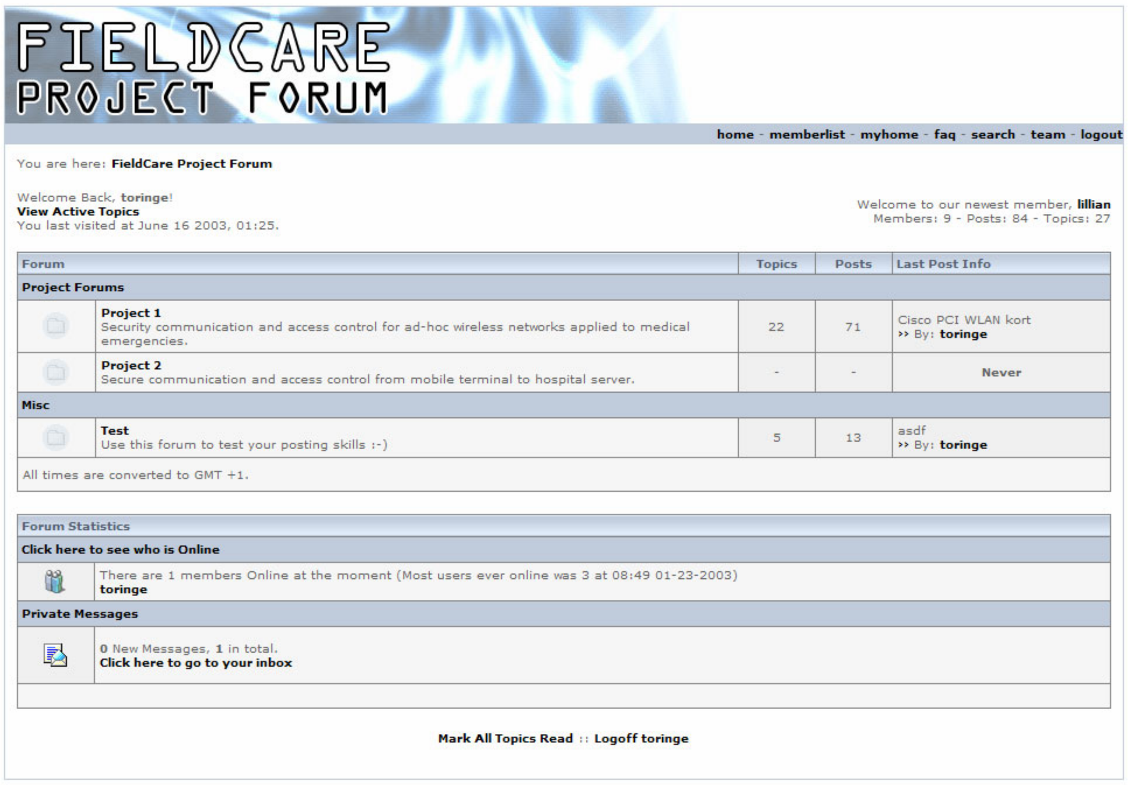
All the Java documentation generated by javadoc is included on the CD attached to this report, rather than inserting it into this appendix. We think that is the best solution, because of the easy-to-use html navigation provided by javadoc. The CD includes the JavaDoc of the entire FieldCare application, the iButton Admin and the InfoServer.

C.2 JAVA SOURCE CODE

We decided not to include our source code in this appendix for practical reasons, as the code would be completely out of context from the rest of the FieldCare application. However the entire FieldCare source code, with our implementations and modifications, are included on the CD attached to this report.

C.3 PROJECT FORUM

We installed the OpenBB discussion forum to be able to properly register all the activity throughout the work with this master thesis, but also to allow our supervisor to get an insight into our implementation. This forum is an open source software solution, and is available at http://www.openbb.com.



You are here: **FieldCare Project Forum**

Welcome Back, **toringe**!
View Active Topics
You last visited at June 16 2003, 01:25.

Welcome to our newest member, **lillian**
Members: 9 - Posts: 84 - Topics: 27

Forum	Topics	Posts	Last Post Info
Project Forums			
Project 1 Security communication and access control for ad-hoc wireless networks applied to medical emergencies.	22	71	Cisco PCI WLAN kort >> By: **toringe**
Project 2 Secure communication and access control from mobile terminal to hospital server.	-	-	Never
Misc			
Test Use this forum to test your posting skills :-)	5	13	asdf >> By: **toringe**

All times are converted to GMT +1.

Forum Statistics

Click here to see who is Online

There are 1 members Online at the moment (Most users ever online was 3 at 08:49 01-23-2003)
toringe

Private Messages

0 New Messages, **1** in total.
Click here to go to your inbox

Mark All Topics Read :: **Logoff toringe**

Figure C.1 – Project Forum
A screenshot of the forum application we used.

An offline version of this forum (with all the registered posts) are available on the attached CD, in the directory `/Project-Forum/offline-version/`, and are viewable in any web browser.

C.4 MISC SCRIPTS

The various scripts constructed to be used in connection with FieldCare and those made only for practical reasons are located in this section.

C.4.1 RESTARTFC

As was presented in section 6.4.4, the logout functionality in FieldCare is simply implemented by shutting down the application `System.exit(0)` and automatically restarting it. Before the application shuts down, a shell script, called `restartFC`, is executed from FieldCare in a separate process. The script has an initial delay of two seconds before it restarts FieldCare.

```
#!/bin/sh
#
# Restart the FieldCare application
#
echo "Please wait while restarting FieldCare..."
sleep 2
java -noverify -jar Fieldcare.jar
```

If the device is either running FieldCare in static mode (`-S`) or bypassing the iButton access control (`-B`), this script has to be modified to accommodate these modes.

This solution is just a hack, due to lack of time, and should later be changed so that FieldCare itself takes care of the logout sequence without exiting the application.

C.4.2 OLSRD

The FieldCare application uses two scripts to initiate and terminate the OLSR daemon. These scripts have to exist and have the proper permissions on the device before running FieldCare. Both scripts should be located in the `/bin` folder and have `744` permissions.

The startup script `/bin/olsrd`:

```
#!/bin/sh
#
# Initiate OLSR Daemon
#
# REMARK: Remember to comment out or delete the
#         line not applicable to your system.
#
# Line 1: For PCs
# Line 2: For iPAQs
#
/usr/sbin/nolsrd –i eth1
/usr/sbin/armolsrd –i eth0
```

The termination script `/bin/killolsrd`:

```
#!/bin/sh
#
# Terminate the OLSR Daemon
#
# REMARK: Remember to comment out or delete the
#         line not applicable to your system.
#
# Line 1: For PCs
# Line 2: For iPAQs
killall nolsrd
killall armolsrd
```

It is possible to have the FieldCare application do all this work, since it is already capable of differentiating between the various device types (MDA and CMDA), and simply executing the shell commands directly. However, again due to lack of time we decided to go for the simplest solution.

C.4.3 INFOSERVER SECURITY POLICY

The security policy file to the secured InfoServer (the one that we have implemented SSL in) is called `InfoServerSecurityPolicy` and is located in the `infoServer/` directory.

```
grant {
      permission java.io.FilePermission "../config", "read,write";
      permission java.io.FilePermission "/usr/java/j2sdk1.4.1_02/jre/
            lib/ext/msutil.jar", "read,execute";
      permission java.io.FilePermission "<<ALL FILES>>", "execute";
      permission java.net.SocketPermission "*:1024-", "accept,
            connect, resolve";
      permission java.net.SocketPermission "localhost:1024-", "accept,
            connect, resolve";
      permission java.util.PropertyPermission "*", "read,write";
};
```

Since the JRE version (Blackdown JRE 1.3.1 for StrongARM) installed on the iPAQs didn't have support for the JSSE, hence no support for SSL, we had to define a separate security policy for the InfoServer where SSL has not been implemented. This file is called `InfoServerSecurityPolicyNoSSL` and is also located in the `infoServer/` directory.

```
grant {
      permission java.io.FilePermission "../config", "read,write";
      permission java.io.FilePermission "/usr/java/j2sdk1.4.1_02/jre/
            lib/ext/msutil.jar", "read,execute";
      permission java.net.SocketPermission "*:1024-", "accept,
            connect, resolve";
      permission java.net.SocketPermission "localhost:1024-", "accept,
            connect, resolve";
};
```

During the testing phase, we used a policy file that grants all permissions. This file should of course only be used when debugging the application.

```
grant {
            permission java.security.AllPermission;
};
```

C.4.4 MANAGING THE WIRELESS CONNECTION

We created two scripts to make the transition between ad-hoc mode and managed mode (i.e. using an access point) and vice versa easier.

The script enabling the adhoc mode: `/bin/adhoc`

```
#!/bin/sh
#
# Enables AD-HOC network mode
#
echo "Initiating Ad-Hoc mode:"
cp /etc/pcmcia/wireless.opts.fcareadhoc /etc/pcmcia/wireless.opts
echo "[!] Please wait while restarting network interface"
/etc/init.d/pcmcia restart
sleep 5
ifconfig eth0 192.168.1.2 netmask 255.255.255.0
```

The script enabling the managed mode: `/bin/managed`

```
#!/bin/sh
#
# Enables managed network mode
#
echo "Initiating connection with access point"
cp /etc/pcmcia/wireless.opts.fieldcare /etc/pcmcia/wireless.opts
/etc/init.d/pcmcia restart
```

C.4.5 RESTORE SCRIPT

Since the iPAQs we used had a very limited amout of storage capacity (only 16MB of non-voliatile memory), we had to install the Java Runtime Environment, the FieldCare application and the InfoServer application in the RAM of the iPAQ. All this would lost when rebooting the iPAQ, therefore we created the following script that automatically downloaded the Java Runtime Environment, FieldCare and the InfoServer from the web server we had installed in the lab. The script had to be modified for each iPAQ, and the one displayed below is for the MDA2 device.

```
#!/bin/sh
#
# Automatically restores the iPAQ
#
echo "RETRIEVING JRE 1.3.1"
mkdir -p /mnt/ramfs/usr/local/jre131
scp -r root@129.241.209.106:/home/project/IPAQ-Linux-apps/jre131arm/.
      /mnt/ramfs/usr/local/jre131/
mkdir -p /mnt/ramfs/home/root/fieldcare
cd /mnt/ramfs/home/root/fieldcare
echo ""
echo "RETRIEVING FIELDCARE MDA2 SOFTWARE"
wget http://129.241.209.106/fieldcare_mda2/config
wget http://129.241.209.106/fieldcare_mda2/Fieldcare.jar
wget http://129.241.209.106/fieldcare_mda2/login.data
wget http://129.241.209.106/fieldcare_mda2/hosts
wget http://129.241.209.106/fieldcare_mda2/med.txt
wget http://129.241.209.106/fieldcare_mda2/HashPin.class
wget http://129.241.209.106/fieldcare_mda2/HashPinExecutor
wget http://129.241.209.106/fieldcare_mda2/restartFC
wget http://129.241.209.106/fieldcare_mda2/start
wget http://129.241.209.106/fieldcare_mda2/MY_IP_ADDRESS
touch buffer.file
mkdir infoServer
cd infoServer
wget http://129.241.209.106/infoServer/.infoClientKeyStore
wget http://129.241.209.106/infoServer/.infoServerKeyStore
wget http://129.241.209.106/infoServer/Client.class
wget http://129.241.209.106/infoServer/ClientSecure.class
wget http://129.241.209.106/infoServer/DBconnection.class
wget http://129.241.209.106/infoServer/InfoServerSecurityPolicy
wget http://129.241.209.106/infoServer/InfoServerSecurityPolicyNoSSL
wget http://129.241.209.106/infoServer/KeepAlive.class
wget http://129.241.209.106/infoServer/nopolicy
wget http://129.241.209.106/infoServer/RMISSLClientSocketFactory.class
wget http://129.241.209.106/infoServer/RMISSLServerSocketFactory.class
wget http://129.241.209.106/infoServer/run.mda2
wget http://129.241.209.106/infoServer/run.mda2.nopolicy
wget http://129.241.209.106/infoServer/run_secure.mda2
wget http://129.241.209.106/infoServer/SecurityConstants.class
wget http://129.241.209.106/infoServer/Server.class
wget http://129.241.209.106/infoServer/Server_Skel.class
wget http://129.241.209.106/infoServer/Server_Stub.class
wget http://129.241.209.106/infoServer/ServerInterface.class
wget http://129.241.209.106/infoServer/ServerSecure.class
wget http://129.241.209.106/infoServer/ServerSecure_Skel.class
wget http://129.241.209.106/infoServer/ServerSecure_Stub.class
mv run.mda2 run
mv run.mda2.nopolicy run.nopolicy
mv run_secure.mda2 run_secure
chmod 755 *
echo ""
echo "CLEANING..."
cd /mnt/ramfs/usr/local/jre131/
rm -rf man/
cd bin
rm policytool
rm keytool
rm awt_robot
rm tnameserv
cd ../lib/audio/
rm soundbank.gm
cd ../images/cursors/
rm -f *.gif
rm cursors.properties
cd ../../
rm sunrsasign.jar
echo "DONE!"
```

The last couple of lines (from “Cleaning..”) removes non-vital (hopefully) elements from the JRE. This was done because of the lack of available storage on the iPAQs.