ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
COLLEGE OF NATURAL SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

Privacy Aware Pervasive Healthcare System (PAPHS)

Etsegenet Gebremeskel

A THESIS SUBMITTED TO

THE SCHOOL OF GRADUATE STUDIES OF ADDIS ABABA UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

IN COMPUTER SCIENCE

April, 2014

Addis Ababa

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
COLLEGE OF NATURAL SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

Privacy Aware Pervasive Healthcare System
(PAPHS)

Etsegenet Gebremeskel

ADVISOR: Dejene Ejigu (PhD)

CO-ADVISOR: Admasu Tenna (Asst. Prof)

Name and Signature of Members of the Examining Board

	Name	Signature
1. Advisor:	Dejene Ejigu (PhD)	_____
2. Co-Advisor:	Admasu Tenna (Asst. Prof)	_____
3. Chair Person:	_____	_____
4. Examiner:	_____	_____

Acknowledgement

First of all, I would like to thank my lord God for his countless love and care to me. I would also like to express my special gratitude to my advisors Dr. Dejene Ejigu and Dr. Admasu Tenna, who have been a tremendous mentor for me. I have taken efforts in this thesis work. However, it would not have been possible without the kind support of my advisors.

I would also like to thank the physicians, nurses and other healthcare providers working in the Black Lion Hospital who helped me to gather the required data for my thesis work. I would especially like to thank my friends working in the same hospital, Dr. Abaynesh Haftu and Dr. Teklay Gebrehawarya for their co-operation and encouragement. I would also like to thank my friend, Gebreslassie Gebrelibanos for providing me very important papers from abroad.

I especially thank to my family for their prayers for me since it was what sustained me thus far. Finally I would like to express my deepest heartfelt to my beloved husband Tsigabu Weldu who was always my support in the challenging moments.

Table of Contents

List of Figures

List of Tables

Acronyms and Abbreviations

AAU	Addis Ababa University
AIDS	Acquired Immune Deficiency Syndrome
ART	antiretroviral therapy
ARV	antiretroviral
BLH	Black Lion Hospital
CAPHS	Context Aware Pervasive Healthcare System for HIV/AIDS Patients
CD4	cell T-lymphocyte bearing CD4 receptor
EHR	Electronic Health Records
HIV	Human Immunodeficiency Virus
MAC	Mycobacterium Avium Complex
PAPHS	Privacy Aware Pervasive Healthcare System
PawS	Privacy Awareness System
PC	Personal Computer
PCE	Pervasive Computing Environment
PCP	Pneumocystis pneumonia
PR	Patient Records
PriVA	Privacy Violation Avoider
RAVE	Ravenscroft Audio Video Environment
TPM	Trusted Platform Module
WHO	World Health Organization

Abstract

In pervasive healthcare environment, patients interact with smart devices and service providers in order to make the system functional. However, patients' information is transmitted via the wireless communication which is relatively unsecure media. Due to the complexity nature of pervasive computing environments, individuals raise security and privacy issues. This paper presents a Privacy Aware Pervasive Healthcare System (PAPHS) which monitors HIV patients remotely. Privacy management techniques are used according to the system's security requirements. This paper also shows how secondary users of medical information can get patients' information towards their need. This paper is focused on balancing patients' privacy concern and information required by the pervasive healthcare system in order to accomplish its task properly by protecting the patients' anonymity. It also focused on balancing the patients' privacy concern and the needs of medical information end users by providing information to them based on the patients' privacy protection policies.

Key Words: Pervasive Computing Environment, Pervasive Healthcare Systems, Privacy, Security, PAPHS.

Chapter 1 : Introduction

1.1 Background

Pervasive computing, which is also referred to as ubiquitous computing or ambient computing, is a post desktop model of human-computer interaction in which information processing has been integrated into everyday objects and activities. Mark Weiser, the father of ubiquitous computing, proposed the term "ubiquitous computing" around 1988. In one of his papers, he stated "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it" [1]. The idea is that technology is moving beyond the personal computer to everyday devices with embedded technology and connectivity as computing devices become progressively smaller and more powerful. Pervasive computing is the result of computer technology advancing at exponential speeds. Pervasive computing goes beyond the realm of personal computers; it is the idea that almost any device, from clothing to tools to appliances to cars to homes to the human body to coffee mugs, can be embedded with chips to connect the device to an infinite network of other devices.

Pervasive computing environments (PCEs) are always mentioned in the context of improving healthcare services. These examples involve user monitoring devices that can upload information to a personal computer (PC) for collection and dissemination to professional caregivers [2]. In the increasingly mobile society the vision of Pervasive healthcare or healthcare to anyone, anywhere and anytime by removing time, location and other restraints could be fulfilled with the worldwide deployment of mobile and wireless networks.

Pervasive healthcare can be defined from two perspectives. As stated by Korhonen and Bardram, it can be defined "first, as the application of pervasive computing technologies for healthcare, health, and wellness management; second, as making healthcare available everywhere, anytime pervasively" [4]. The term pervasive healthcare has originated from the integration of pervasive computing in healthcare. The need for pervasive healthcare systems basically originate from the following two facts: (a) healthcare professionals experience a high level of mobility and they need to share resources and information with the staffs, faculty and colleagues in real time and (b) there is an increasing trend in favor of treating chronic disease or recovering patients at

home. In such cases, there is a critical need for ubiquitous access to data and resources for research, diagnosis, emergency treatment and so on [5].

1.2 Motivation

It took some time to get a majority of physicians in the U.S. to agree that it would be beneficial to implement electronic health records in their practices. Now, a survey finds the most skeptical audience for EHRs are patients. According to the survey made by Xerox of more than 2,100 patients, only 26% want their medical records to be digital. Only 40% believe that electronic health records will result in better and more efficient care. And 85% of them stated their concern on the privacy and security of their information [61].

Pervasive computing is one of the most promising problem solving technologies emerging in the modern world especially in healthcare applications. One of the fundamental properties of pervasive computing systems is the ability to adapt to the needs of each user which is commonly known as context awareness. Context information is produced by tracking the actions and collecting real time user data, such as heart rate, body temperature, location and so on. As the pervasive system collects more information about its users, the quality of its context aware services increases, and is a potential threat to the users' privacy if the context information is compromised.

Hippocrates, the father of medicine, already realized that "the physician must not only be prepared to do what is right himself, but also make the patient ...cooperate" [3]. Patients can get healthcare services via the PCE but privacy issue is the major bottleneck for the patients as well as the medical staff. Privacy issues of individuals are commonly discussed among researchers, end users and practitioners in the pervasive healthcare environments. Although pervasive healthcare systems are applications that can support patient's need anytime and anywhere, it raises privacy concerns since it can lead to situations where patients may not be aware that their private information is being shared and becomes vulnerable to threats.

In pervasive healthcare settings, protecting the privacy of patients and the medical staff is very critical, because lack of privacy may hinder the broad acceptance of pervasive healthcare technology. Moreover, there is a need of empowering end-users with flexible personalized controls for their personal data collected, processed and communicated through the pervasive

healthcare infrastructure. We illustrate scenarios from pervasive healthcare settings where privacy violation is a concern.

Scenario_1

As shown in Figure 1-1 [44], there is a healthcare monitoring system with three privacy concerned stakeholders. These are: patient, family and healthcare providers. The patient is an elderly man who is the father of a working son. And his father wants to live alone in his house and the son would like to monitor his father from his office. The monitoring system provides cameras in the house to monitor the patient's movement and allows the son to watch the video. The healthcare providers are not allowed to watch the video at any time except in case of emergency in order to protect privacy of the patient. When the son sends emergency notification, healthcare providers can get the patient related information in order to send ambulance and provide treatment. This kind of restriction can protect privacy of the patient.

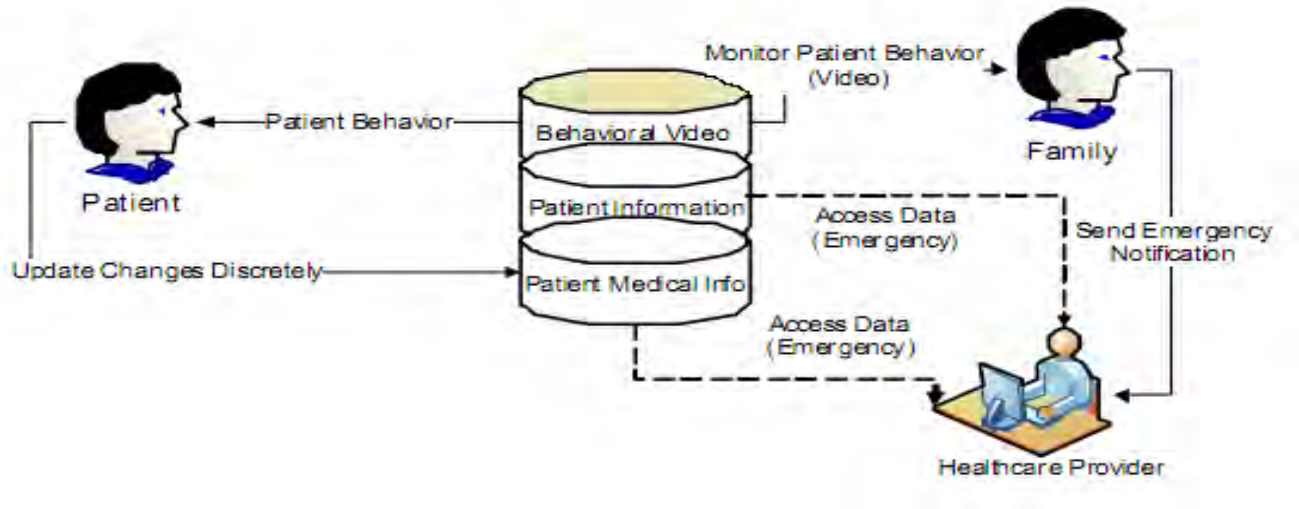


Figure 1-1: *Pervasive healthcare monitoring system overview*

Scenario_2

A young boy is suffering from high blood pressure and recently has experienced a minor stroke too. He always prefers to be at home after he was released from the hospital. His doctors who treat him think that he is not free from danger so he should be under their supervision for some time. The boy has a central server at his house where a healthcare monitoring system is installed in order to monitor his daily activities and physiological parameters. He is under the care of the

physician, nurse and psychiatrist via the pervasive healthcare system installed that gathers data about the patient, interprets and draw conclusion about his health status. When the patient is suffering of a problem, the system realizes and notifies his doctors and relatives. The patient is the owner of the information in the system and has the right to grant or deny others to have access to the data. Below are scenarios showing cases for privacy concern:

a) The patient does not allow sharing his medical records with others except with his treating physicians. He owns the past and recent information in his personal record (PR), but his doctors own only his recent medical information. The patient is highly concerned about the capability of his visitors to access his PR since his pervasive system allows anyone entering his house with a new pervasive device could possibly gain access to his sensitive information if it is not secured. It is very difficult to protect the medical information as it is not clear who the owner of this information is and who has the right to grant access. Due to the security hole found in the pervasive system in his home, he wants the system to enforce users' authentication and access control policies in order to ask approval before granting any access to his information.
b) One of the doctors who treat the patient is the owner of the blood pressure measurements and prediction information. Although the owner of the information is the patient, the doctor does not want the patient to access the medical information and thus the doctor has the right to keep the information secret for treatment purpose. In some cases this can be considered as violation of revealing the information.
c) The nurse, who is assigned to take care of the patient, needs to access his medical history but the nurse has not yet been granted access to view his information. To provide access right to that nurse, the doctor should be at the patient's house. But the doctor does not want to be known by the nurse that the doctor is at the patient's house. This kind of information disclosure is because of the implications drawn based on the context that the nurse gaining accesses to the patient's information infers the presence of the doctor at the patient's house. This context information violates privacy of the doctor.

1.3 Statement of the Problem

People are depending more and more on handheld devices such as cell phones, smart phones, tablet personal computers, Personal Digital Assistants etc. and these are becoming an integral part of human life. The technological advancement in the fields of handheld devices, wireless networks, sensors and actuators has made it possible to work in the PCE [6]. Pervasive healthcare is one application of pervasive computing which aims to enhance patient independent living and the quality of life and pay special attention to issues such as ease of use, privacy and security. Wireless network infrastructure is classified into various categories as in wired networking. However, wireless networks have been relatively unsecure due to lack of technological support and improper configuration. From its very nature of being open and dynamic, PCE has been challenged with privacy and security related issues. Unless the security and privacy issue of pervasive healthcare systems are properly solved, it can hinder the broad acceptance of such systems.

In Ethiopia, most of the patients are afraid of telling their medical cases for others due to different reasons. Some of the reasons are:

- Big hospitals like Black Lion Hospital are crowded by patients every time. It is also true that most of HIV patients do not feel free to go to the hospitals periodically due to fear of social discrimination. Such problems can be solved by controlling the patients who have chronic diseases such as HIV using pervasive healthcare systems remotely. However, such systems need security and privacy protection mechanisms. Therefore, for countries like Ethiopia, privacy concern of patients can hinder the acceptance of pervasive healthcare systems in the countries.
- In Ethiopia, due to the existing manual medical records management system, patients' information is exposed to different security threats.
- It is also true that medical information is shared among physicians and other users. For example, in some cases, end users of medical information may need medical information with clear identity of the patients. However, physicians may take a general conclusion on protecting privacy of the patients. Thus, they may encrypt the names of all the patients before providing the information to the end users. Such conflicts of interest always happen.

- In the existing manual system, physicians give codes for identity indicator attributes of patients during provision of medical information. However, the codes given for a particular patient luck consistency since the code given for that patient may differ at different times.

1.4 Objectives of the Thesis

General objective

The general objective of this thesis work is to design and develop a dynamic medical information access control model and the data sharing mechanisms based on the owner's consent for the pervasive healthcare systems.

Specific objectives

a. In depth identification of issues and parameters that address privacy concerns in pervasive healthcare environment.
b. Designing of access control model based on the identified parameters.
c. Enhancing the designed access control model with mechanisms that allow patients to dynamically set level of confidentiality on their own medical records to be accessed by the care givers and other users.
d. Design data sharing mechanism that governs provision of patient medical information to others only based on patients' consent. It also keeps consistency of a privacy protection code given for a particular patient through time in order to help the end users of medical information during data analysis.
e. Develop a prototype and integrate the designed privacy protection system with selected pervasive healthcare system.

1.5 Scope and Limitation

This thesis work has its own scope and limitations. The scope of this thesis work is on the design and development of privacy protection mechanisms for pervasive healthcare systems. The access control mechanism will be developed based on the privacy management techniques which are commonly used in the pervasive computing environment and are suitable for the selected pervasive healthcare system.

This thesis work has the following limitations:

- The pervasive healthcare system that we use to test our main work (privacy protection) is implemented by prototype because of the limitation of the allocated budget.
- The pervasive healthcare system only monitors HIV patients using CD4 cell counter device.

1.6 Methodology and Tools

The methods and tools that will be used in this thesis work are stated below.

Methods

1. **Literature review:** related works will be reviewed in order to have a deeper understanding on privacy concerns of individuals in pervasive computing environments. A review on different privacy management techniques in pervasive computing environments will also be made. A review on protecting privacy issues in pervasive healthcare environments will also be made.
2. **Interview:** we will have an interview about privacy issues with inpatients and outpatients. This will be conducted on HIV patients who are the most privacy concerned ones as mentioned on other surveys results.
3. **Questionnaire:** we will prepare and distribute questionnaires among the healthcare givers (physicians and nurses) in order to know the extent of patients' privacy concern from their point of view.
4. **Onsite observation:** we will observe additional information during the interview that will be made with patients. We may also have some feedbacks from healthcare providers while distributing and collecting the questionnaires.
5. **Prototype development:** in order to proof the concept of this thesis work, we are going to develop a prototype of the system using appropriate tools.
6. **Evaluation:** after the prototype is developed, it will be tested based on some scenarios that will be prepared for this purpose. The prototype of the system will be evaluated by comparing with the privacy management techniques used in both the pervasive computing environments and the manual systems.

Tools

1. **J2ME platform SDK:** we will use this tool in order to emulate the mobile devices and make a wireless communication with the remote healthcare server that controls the patients remotely.
2. **Apache Tomcat server:** we will use this server in order to run the web applications that control patients' health status remotely and protect unauthorized access of medical information.
3. **MySQL database server:** we will use MySQL database in order to store medical information of patients and privacy protection policies set by the patients.
4. **SPSS software package:** we will use SPSS tool for analysis of the data that will be collected using the data gathering techniques mentioned above.

1.7 Application of Results

The output of the research has the following applications in different areas:

1. **Patients:** the system will help patients to control their privacy sensitive health records collected through time in pervasive healthcare environment. It can be also implemented on one of the most recent pervasive healthcare systems "Context Aware Pervasive Healthcare System for HIV/AIDS Patients (CAPHS)" [28], where security is one of the future works of this thesis work.
2. **Medical staff of health centers:** the system enables them to decide what kind of patient information should be provided to a particular user request (Employer, Insurance Company, Court of Law, Medical Research, etc). It also helps them to monitor their patients by getting their status remotely in a secure manner.
3. **Business centers:** it is commonly known that most system developers focus on the development of the systems than the security issues of its end users. The output of the research can create awareness on the crucial issue of privacy on the design and development of pervasive healthcare systems.
4. **Researchers:** the result of the research can help as a reference for researchers who work in the areas of privacy protection of individuals in the pervasive healthcare environment.

5. **Users of medical information:** the system helps for the users of medical information (Medical researcher, public policy makers, public health management etc..) in getting the data according to the owners' consent. It also satisfies the need of the owners of medical information in protecting their data. Some patients may be privacy concerned and others may not worry about privacy protection on a particular disease. However, for the end users, it is advantageous than the data provided by physicians based on their general conclusion made to protect patients' privacy. The system also helps medical researchers by giving the same code for the house number of patients who live in the same house and the code given for the name of a particular patient at different times is the same. In some cases such kinds of information is very important for medical researcher during analysis of data.

1.8 Organization of the Thesis

Including this chapter the document has a total of seven chapters. ***Chapter two*** gives brief details on the concept of personal privacy, privacy issues in PCEs especially in pervasive healthcare system, privacy issues of patients from the medical ethics point of view and the privacy management techniques used in PCEs. ***Chapter three*** presents several research works in protecting security and privacy of individuals in PCEs and pervasive healthcare systems. ***Chapter four*** discusses on the findings and lessons learnt from the survey made on privacy related concerns of patients from the stakeholders' point of view for the case of Black Lion Hospital. ***Chapter five*** presents the proposed Privacy Aware Pervasive Healthcare System (PAPHS). It briefly describes architecture of the system followed by detailed explanation of each component of the system. ***Chapter six*** then gives details on the developed prototype implementation named as PAPHS based on an example scenario followed by the demonstration. The document ends with discussion on the achievement and conclusion of the thesis work and presentation of the planed future works.

Chapter 2 : Literature Review

In this Chapter, we first introduce what personal privacy is, privacy concerns of individuals in the PCEs and especially in pervasive healthcare systems. We also explain privacy issues of patients from the medical ethics point of view. This chapter ends with discussing on the commonly known privacy management techniques on PCEs.

2.1 Overview on Personal Privacy

Privacy has been on people's mind as early as the 19th century, when Samuel Warren and Louis Brandeis wrote the influential paper "The Right to Privacy" [26], motivated largely by the advent of modern photography and the printing press. While Brandeis defined privacy as "the right to be let alone" (arguing against nosy reporters who would take pictures of people without permission – previously one had to sit still for a substantial amount of time, otherwise the picture would be all blurred), most people nowadays think of it more as "the right to select what personal information about me is known to what people" [27].

2.2 Privacy Issues in Pervasive Computing Environments

Since the earliest days of research in pervasive and ubiquitous computing, preservation of user privacy has been the subject of an active discussion [7, 8, 9]. On the one hand, a PCE needs to collect a large amount of information about its users to be able to adapt and respond to their demands without requiring much explicit user interaction. On the other hand, the more information a system has about its users, the greater is the potential threat to the users' privacy. It is a core design feature of most PCEs to preserve the anonymity of the clients (users, services, or smart devices) interacting with service providers. However, an inherent contradiction lies in the fact that the service provider cannot fully trust the client that does not reveal its true identity. Without such a trust, the service provider cannot ascertain that the client has a rightful access to the requested resource or service. As a result, there is a perceived conflict between privacy and the technologies that enable the features of a PCE dealing with sensing and storing of user-related information. Personal privacy related to information about the user's identity, past and current activities, and location, may be challenged because these kinds of information are typically among the data collected and stored by the sensors and services employed by most PCEs [10].

Research literature provides a wide range of definitions of privacy in PCE: "control over information disclosure" [11], "privilege of users to determine for themselves when, how, and to what extent information about them is communicated to others" [12], and "ability of an individual to control the terms under which their personal information is acquired and used" [13]. A detailed discussion of many broad aspects of privacy in various contexts can be found in [14, 8, 15, 16, 17]. In PCEs, users may find an increasing amount of obstacles to maintain their privacy because of the growing volume of potentially identifiable data collected by the systems [18]. Data collected by perceptual interfaces that are capable of recognizing users, their gestures and facial expressions, may become a potential threat to the user privacy if it is somehow made accessible to a third party. Finally, some users may be fearful of compromised privacy and not be willing to accept and use a system that tracks and collects information about them, even though this data by itself may pose no privacy threats [18]. While there is a significant amount of research in pervasive computing aimed at design and implementation of privacy management tools and techniques, their practical usability and acceptance remains an important challenge [19]. In 2003, Computing Research Association identified the ability to "give computer end-users security they can understand and privacy they can control" in the "dynamic, pervasive computing environments of the future" as one of the four major research challenges of trustworthy computing [20].

Recently, a number of studies explored social implications of pervasive technologies and their impact on the changing perceptions of privacy [21, 22, 23]. As the technologies enabling PCE become ever more miniaturized and further blend with the surroundings, the users may not recognize or feel their presence and lose the awareness of the fact that some or all of their actions may be monitored and recorded by a PCE. Continuous collection of information about the users will inevitably expose the details of their personal traits: patterns of behavior, driving and walking routes, current location, shopping preferences, likes and dislikes, social associations, etc. Most studies suggest that some users may be willing to expose certain information about their behavior and actions in order to enable the adaptive nature of a PCE; at the same time, they may prefer to establish some boundaries and let other aspects of their activities remain private. Users also may prefer to have the tools to limit the amount or granularity of the collected information, or to stop the monitoring altogether if they decide to do so [21]. Although the very notion of

pervasive computing suggests that the technology should become invisible and blend into the environment, some users prefer to know that the technology actually is there [22].

2.3 Privacy Issues in Pervasive Healthcare Systems

Among many real world application domains where pervasive computing systems have been implemented, healthcare is one area in which privacy plays the most prominent role [24, 25, 5]. Hospitals, clinics and emergency rooms are a perfect test bed to implement a PCE: information services provide real-time data about patients, their medical history and treatment records; this information must be treated with confidentiality, yet available anytime to authorized users. Healthcare environment requires a high availability of information services, constant coordination among colleagues, and rapid response to emergencies [5]. A healthcare-oriented PCE must support a high degree of mobility for its collaborating users, who must have real-time access to the patient data protected by strong privacy safeguards. Depending on the specialization of a medical professional and their role in the treatment of a given patient, they may have access to different areas of records of that patient. Implantable medical devices used in a healthcare-oriented PCE present an even wider range of privacy-related challenges that are unique to this domain [24]. The other open issue of the CAPHS is also security and privacy of patients during transmitting their vital sign information [28].

2.4 Medical Ethics

Unlike law, ethics advises higher standards of behavior. Laws can differ from country to country while ethics is applicable across national boundaries [45]. Medical ethics is related to law that specify how physicians are required to deal with ethical issues in patient care and research. Confidentiality is essential to the trust between patients and physicians. It is also a fundamental principle of medical ethics. Confidentiality is a critical issue since human beings deserve respect from others. Preserving privacy of individuals is one way of showing respect to them. Since individuals can have different privacy concerns, care must be taken by determining which personal information a patient needs to keep hidden and on which he/she is willing to share with others. The World Medical Association's (WMA's) Code of Medical Ethics states, "It is ethical to disclose confidential information when the patient consents to it or when there is a real and imminent threat of harm to the patient or to others and this threat can be only removed by a breach of confidentiality" [45].

Physicians have the duty to keep accurate and up-to-date patient medical information. They are also responsible to treat their patients with respect for their dignity. They have to act in the interest of patients while providing medical care since the reverse way might cause weakening the mental and physical conditions of the patient. They are expected to be aware of their obligations under the Data Protection Acts [62] in relation to secure storage and eventual disposal of such information as well as relevant published Codes of Practice [63]. Patients are eligible to get a copy of their own medical information. This right of access is provided by law [64].

There are also cases where medical information is required by health protection staff (Physicians, nurses, laboratory technicians) in order to protect the public. Clinical audit and quality assurance systems are important to provide good care and requires reliable patients' data. Education and training of health professionals are also important to provide safe and effective healthcare. However there can be possible benefits to the public but may harm patients. Proposals for medical research on human must obtain approval once the researchers explained the purpose and methodology of the work. Ethics committee approval [45] is necessary to determine whether a project is scientifically and ethically appropriate. Unlike in the clinical care, research works [45] needs the disclosure of medical records to the wider scientific community and the general public. In order to protect privacy of individuals, researchers must obtain consent from patients (research subjects) before using their medical information for research purpose. It is also true that the role [45] of physicians and researchers with respect to the relationship they have with patients is different, even if the physician and researcher belong to the same person. When medical information is used for Clinical and/or educational purposes it may not be anonymised appropriately or impossible to be anonymised, so that patients should be aware of their medical information disclosure. If there is any objection of the disclosure, it must be respected. Physicians may also be requested by lawyers (legal representatives) or insurances for medical reports of a patient treated professionally. However the reports should not be prepared or given without the patient's permission. Any kind of recordings (Audio, visual or photographic) of a patient or any of his/her relatives should not be disclosed without their consent or should be kept secretly as part of their record.

Patient medical records should remain confidential even after death [46]. If there is no clear consent made by the dead person about his/her information disclosure, they should consider how

the disclosure might benefit or cause distress to the deceased's family or other patients. Individual decisions in this case might be limited by law [65].

Physicians should ensure that the patient's privacy is maintained at all times and compliance with data protection legislation [62]. Even if the patient does not consent to disclose, the physician should respect except where failure to disclose would put others at risk of death or serious harm. Medical information disclosure may be mandated [47] by law in certain limited circumstances such as:

- When there is an order by a judge in a court of law.
- Where required by infectious disease regulations [66].

But in these instances, physicians should inform the patient and should give reasons for it. Once the disclosure of the patient records is justified, carefully consider whether anonymisation of the information (without revealing the patient's identity) can provide the same potential benefits and should be disclosed to the minimum that is only required in the instance. If the medical information is disclosed for the purpose of public protection, Physicians can report to the appropriate health authority and/or the relevant constitutional agency if any serious harm is happened on a patient [47].

2.5 Privacy Management Techniques

In order to overcome the hindrance to the broad acceptance of pervasive computing systems in the world community various techniques are proposed. According to the survey made by Kurkovsky, Rivera and Bhalodi on the techniques that are used in different pervasive architectures [48] : in this subsection, we briefly describe the commonly known privacy management techniques that are used in the design and implementation of some existing pervasive computing systems and architectures.

1. Access rights and policy management

Users in the pervasive computing environment may interact with various service providers and devices to get their services. Since these services and devices could be malicious, the user privacy might be compromised. PerGym [34, 35], a pervasive gym system which offers personalized services depending on privacy sensitive context. Designing Advanced network Interfaces for the Delivery and Administration of Location independent,

Optimized personal Services (Daidalos) [36, 37, 38] provides Privacy Negotiation Manager.

2. Classification of resources

In a Pervasive environment that supports resource sharing, unwanted parties could retrieve confidential information causing information leakage and leading to user privacy violation. Mobile devices carried by an individual user of a pervasive computing can contain a large amount of information where not all of them may be sharable.

3. Data persistence control

In pervasive systems, information can be automatically gathered and stored through time. Due to the privacy concern of individuals in PCE, tools are required to control the disclosure of collected and stored data. The data can be controlled by placing time constraints on the data as defined by either the owner of the information or a service provider.

4. Granularity awareness

Information such as time or location which is accessible in a PCE can be grained down for privacy protection purpose. For example in the case of location, information can be grained down by only asking for the building name instead of the building and a room number. There is also a clear need to maintain the balance between the reasonable resolution of information the users are willing to provide and the amount of information required by a particular service in a pervasive computing environment.

5. Constraints and Permissions

This technique plays an important role in controlling access to services and resources in pervasive computing environments. Role Based Access Control (RBAC) is one of the most widely used methods to control access to services and resource. In this method users are associated with roles which in turn are associated with a set of permissions. An access rights constrained PCE often provides a means to balance the tradeoff between the amount of privacy a user wants to allow and the value of the service the pervasive system will provide to the user. Context Models for Privacy [43] manages the set of constraints and permissions based on facts and situations.

6. Ownership of context information

Based on contextual information acquired in a PCE, information is marked in a way that allows an association with an individual's action. Such association enables the system to protect data by using rules based on the context, which can be also used as additional information while evaluating and interpreting the privacy rules.

7. Information flow obfuscation

Obscuring the actual nature of given information through a system is another technique of privacy protection in a PCE. The disclosure of any information to a third party would be considered as user privacy violation. Information flow can be obscured by employing one or more of the following methods: Concealing the existence of the services available from the service providers, Concealing any information necessary to request services for the service providers, Concealing any information concerning a service accessed by a client, Concealing the exchange of contextual information between the client and the service provider and so on.

8. Service access protection

Sensor technologies may provide context data such as location, health related parameters, time, user activities, behavior, etc which can be used by a PCE. In such cases users may be concerned about the leakage of their private information and thus they may want to know who can access that private data as well as what kind of data are being collected and how that data is being used.

9. Information disclosure protection

In order to protect information disclosure in a PCE, service providers must only get the necessary data to perform its task and at the same time the user should only have to provide that limited amount of information. It is also required to decide on what data elements are to be disclosed, in what format and for what purpose.

10. Protection of information usage

Controlling the dissemination of information is another technique of protecting users' privacy issues. In pervasive systems, information can be collected and automatically stored for various purposes. If a data is stored for future use, it must be protected from access by

any unauthorized users and unintended service providers as well. When there is a request for data the service provider must only release to the intended users only. If a user of a PCE does not have control over the dissemination of their data, privacy of individuals cannot be ensured so that collecting and storing of data will be a source of privacy violation. For such reasons PCE must provide restrictions on information usage by users and service providers.

The techniques chosen for the proposed pervasive healthcare system are very similar to the techniques used for Privacy Awareness System (PawS) [31]. In the next section we will present the related works we have reviewed.

Chapter 3 : Related Work

In this Chapter, we will present some of the papers reviewed which are related to our thesis work. In each subsection the techniques used in each system will be explained briefly. Finally, summary of these related works will be presented in the last subsection as shown in Table 3-1.

3.1 Pervasive Trusted Computing

This paper shows that Trusted Computing can provide a number of useful services in pervasive environs. In some cases, Trusted Computing can be seen as a stable force of homogeneity by providing standardized security interfaces through which heterogeneous devices can interact with each other. In this respect, they have proven its applicability in providing: secured storage of confidential information, anonymous authenticated access to a secure facility, non-revelation of confidential information and an anonymous voting scheme in PCE [30].

The Trusted Platform Module (TPM) is a hardware security solution for mobile devices in a PCE. It implements access rights without revealing identities of its users. It can also authenticate a mobile user by issuing challenges to the device that is then signed by the access requestor. TPM can provide ownership by locking the information using a non-migratable key. Therefore, the owner of the information can unlock it as shown in Figure 3-1 [30]. The privacy management techniques used in this architecture of the system are:

- Access rights management
- Classification of resources
- Ownership of context
- Obscured information flow
- Service access protection
- Information disclosure protection
- Protection of information usage

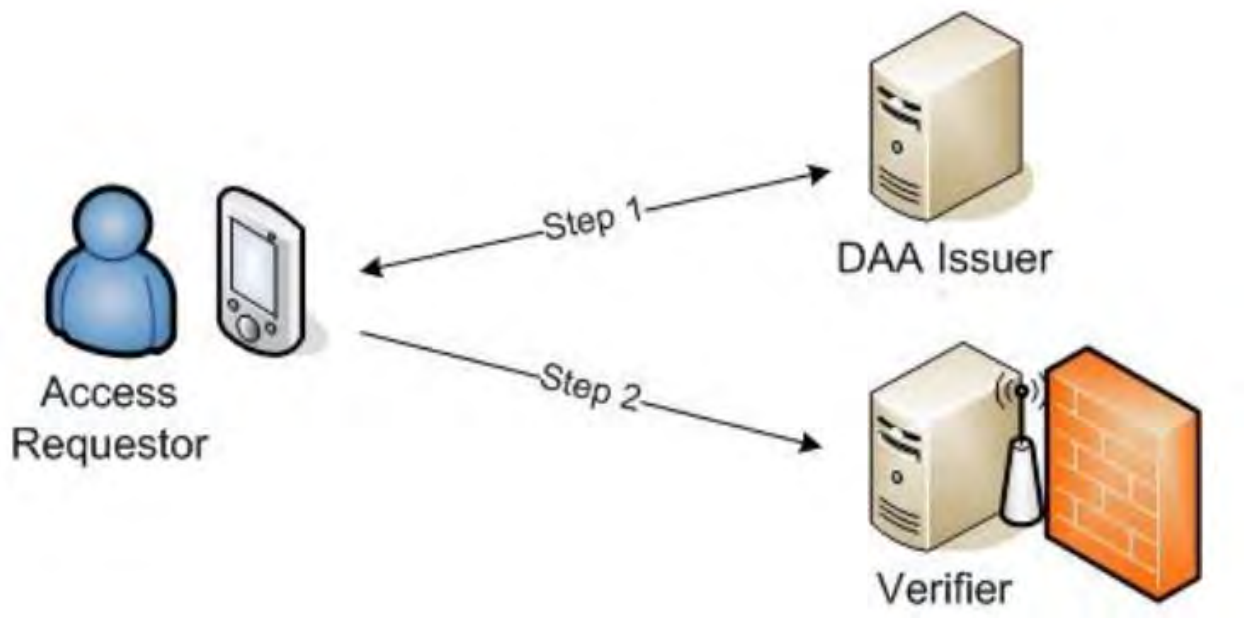


Figure 3-1: *Enrollment and Verification Architecture*

3.2 Spontaneous Interaction in Virtual Multimedia Space: EuroPARC's RAVE system

People can have several opportunities to communicate informally if they work together in the same place. While working together they may share different interests and develop a sense of community. But when people are separated geographically, most of the informal knowledge about each other will not be there and their communication becomes much more formal. The author has developed tools to support people who work together on designed problems but they are geographically distributed. They have preferred to explore the concept of a media space which includes participants with monitors and video cameras in their offices and can select a variety of ways of viewing others or being viewed by others. The system includes video cameras, microphones, speakers and monitors in order to help people at different location to work together remotely. Unlike most of networked videos (video conferencing and video phones) that support focused collaboration, Ravenscroft Audio Video Environment (RAVE) media space can set up long term connections between offices or can also support glance informally at each other. They can keep the shared awareness of others in the building or can use a system known as portholes to keep in touch with the people in the lab on another area [29].

Each user of the system can control who can connect to them and in what kind of connection should it be. RAVE makes its users to be aware of the technologies in use and how their privacy is protected. RAVE has a technique to assign ownership of context and allows them to control their audio-video content and the placement of all the audio-video equipment. RAVE also has a component called Godard which serves as a mediator for all service requests in order to control the dissemination of the audio-video contents [29]. The privacy management techniques used in this architecture of the system are:

- Access rights management
- Access policy management
- Constraints and permissions
- Ownership of context
- Protection of information usage

3.3 Protecting Users' Anonymity in Pervasive Computing Environments

In this paper they have presented the state-of-the-art k-anonymity techniques for privacy protection by anonymizing service requests which are insufficient when applied to many pervasive computing scenarios. To show that, they have formalized shadow attacks and proposed defense techniques which have been experimentally evaluated in a simulated environment. Some of the most relevant issues they are currently investigating are: the definition of a comprehensive measure of privacy, an extension of their privacy protection techniques to support multidimensional k-anonymity for context-aware services, and an extension to support the dynamic case, i.e., when an adversary is able to reconstruct the sensitive association by means of requests issued by the same user in different time intervals [35].

PerGym is a pervasive system which is used for gym. It provides personalized services based on privacy sensitive contexts and aggregates the context data from distributed sources. It has context aware privacy module whose main duty is to enforce the system wide compliance with a set of privacy policies set by its users. The other important duty of the context aware privacy module is to transform service requests coming from each user and make them indistinguishable from those of others. It anonymizes all coming requests by generalizing the context information in the request and by modifying the actual identity of the users [35]. They have presented a particular

example as shown in Figure 3-2 [35]. The privacy management techniques used in this architecture of the system are:

- Access rights management
- Access policy management
- Granularity awareness
- Obscured information flow

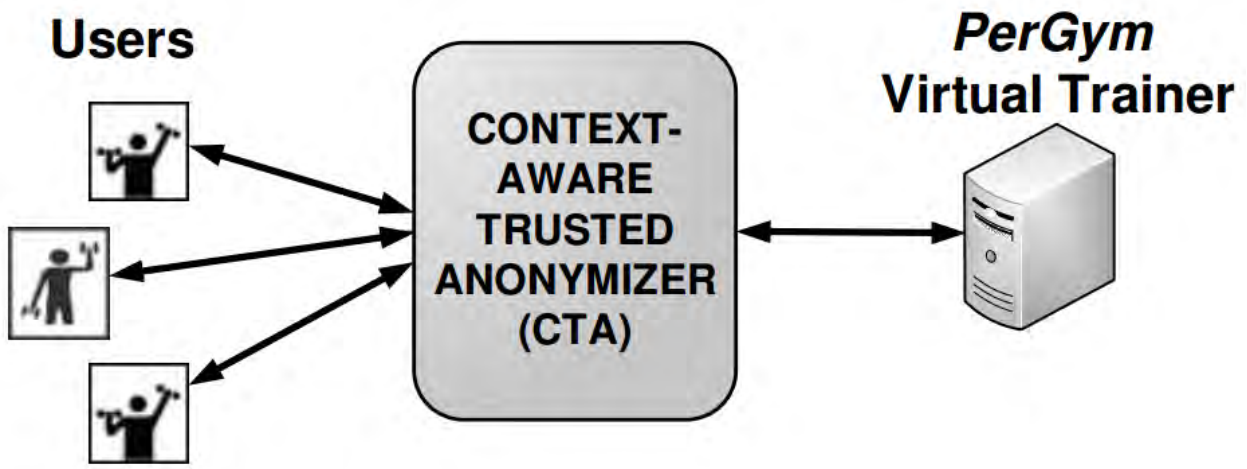


Figure 3-2: *The PerGym Scenario*

3.4 Using User Preferences to Enhance Privacy in Pervasive Systems

There is an increasing trend in developing pervasive computing technologies and fear of users' personal privacy violations due to the problems of maintaining users' privacy. The Daidalos pervasive system (i.e, European research project) hides the real identity of individuals using a system of virtual identities in order to provide privacy protection via pseudonymity. Although Daidalos primarily achieved that, one problem with this lies in determining to what extent the user should be involved in making decisions related to the virtual identities selection and what can be done automatically. This is solved by creating a set of user preferences in order to assist users in taking their decisions and refining them through the use of machine learning techniques. Designing Advanced network Interfaces for the Delivery and Administration of Location independent, Optimized personal Services (Daidalos) is privacy enabled infrastructure in a PCE. Daidalos provides a Privacy Negotiation Manager for personal information of the users [37]. The privacy management techniques used in this architecture of the system are:

- Access rights management
- Access policy management
- Classification of resources
- Service access protection
- Information disclosure protection

3.5 Modeling Privacy Control in Context-Aware Systems

In this paper, they have defined a theoretical model for controlling personal privacy in context aware systems based on a core abstraction of information spaces. They have previously focused on deriving socially based personal privacy objectives in PCE. They aim to use information spaces to create a model for controlling personal privacy which supports their socially based privacy objectives. They also discussed how they can introduce decentralization which is the desirable property of many pervasive systems, into their information space model, using unified privacy tagging. Information Space Model allows its users to define a set of permissions for access to their information, service and resources. It uses information boundaries in order to control the personal privacy of its users by enforcing permissions defined by the owners of each information space [42]. The privacy management techniques used in this architecture of the system are:

- Granularity awareness
- Constraints and permissions
- Obscured information flow
- Information disclosure protection

3.6 Context-Aware Access Control: Making Access Control Decisions Based on Context Information

In PCE access control decisions have to be adaptable to changes in order to make some adjustments to these changes accordingly, instead of using manual interactions. The solution to this change is a context aware access control mechanism where the stated changes are influencing access control decisions. In this paper they presented the design and implementation of a generic and flexible security framework that is used for mobile and ubiquitous business applications. It is implemented for MOSQITO (i.e, European research project). The main

components of the framework are mechanisms that implement security at message level and modules for context information acquisition and trust evaluation processes. These two components are joined together to provide a secure context information exchange and to make security decisions. MOSQITO provides context aware access policy management that is aimed at evaluating trust among the service provider and the consumer in the PCE [33]. Figure 3-3 shows overview architecture of the system [33]. The privacy management techniques used in this architecture of the system are:

- Access rights management
- Access policy management
- Constraints and permissions
- Information disclosure protection

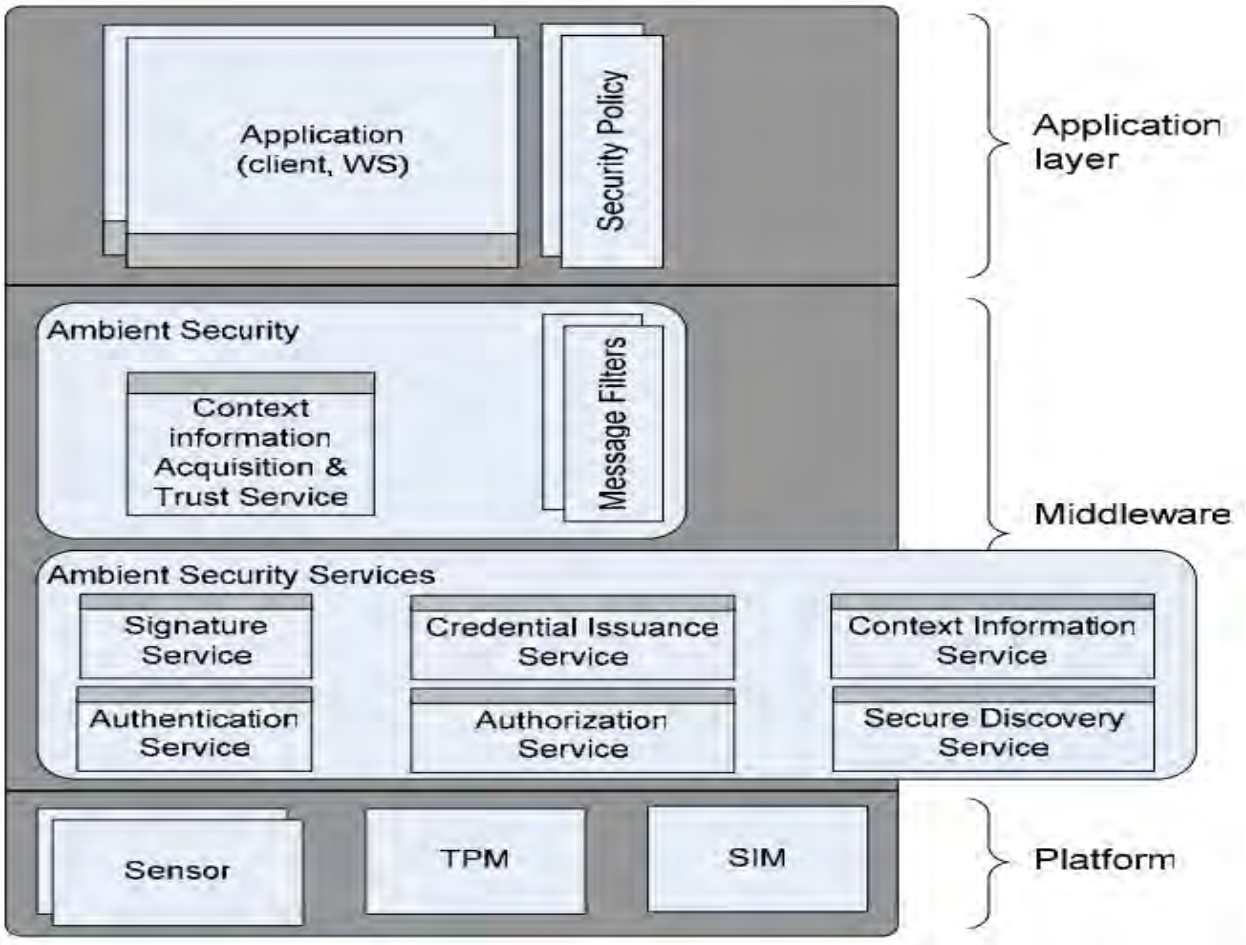


Figure 3-3: *MOSQUITO Framework Architecture Overview*

3.7 Exploiting Hierarchical Identity-Based Encryption for Access Control to Pervasive Computing Information

In pervasive computing environments access control to confidential information is challenging for various reasons. Due to the shortcomings of existing access control schemes that rely on either clients presenting a proof of access to a service or services encrypting information before providing the information to a user, they proposed a proof based access control architecture. It works based on hierarchical identity based encryption in order to enable services to inform users of the required proof of access in a hidden way without revealing their information. They also introduced an encryption based access control architecture that uses hierarchical identity based encryption in order to deal with multiple, hierarchical constraints on access rights. They implemented the proposed architectures in a PCE. Their evaluation shows that identity-based encryption is expensive. However, the overhead can be significantly lowered using a more optimized implementation. Hierarchical Identity-Based Encryption implemented granularity aware technique in order to hide the detail of the information that will be sent to the receiver of the context information in the PCE. The level of granularity of the information depends on the privacy concern of the owner. It also restricts the lifetime of access rights given to each user [40]. The privacy management techniques used in this architecture of the system are:

- Access rights management
- Classification of resources
- Data persistence control
- Granularity awareness
- Constraints and permissions
- Ownership of context
- Obscured information flow
- Information disclosure protection

3.8 A Capability-Based Privacy-Preserving Scheme for Pervasive Computing Environments

In pervasive computing environment, users may interact with various service providers or smart devices in order to get services. But these service providers can be malicious which leads to

users' privacy violation. In addition to this, user authentication is also required for service providers in order to provide service to only authorized users. In order to preserve user privacy, they must be allowed to have anonymous interactions with the service providers which is challenging task for authenticating such users. This paper presents a scheme that allows users to anonymously interact with the service providers and the service providers can effectively authenticate and authorize its users based on their anonymous information. The Capability-Based Privacy-Preserving Scheme also provides a time to live attribute in order to place time constraints on the stored data as defined by the owner or the service provider [39]. Figure 3-4 shows architecture of the system [39]. The privacy management techniques used in this architecture of the system are:

- Access rights management
- Classification of resources
- Data persistence control
- Granularity awareness
- Constraints and permissions
- Obscured information flow
- Service access protection

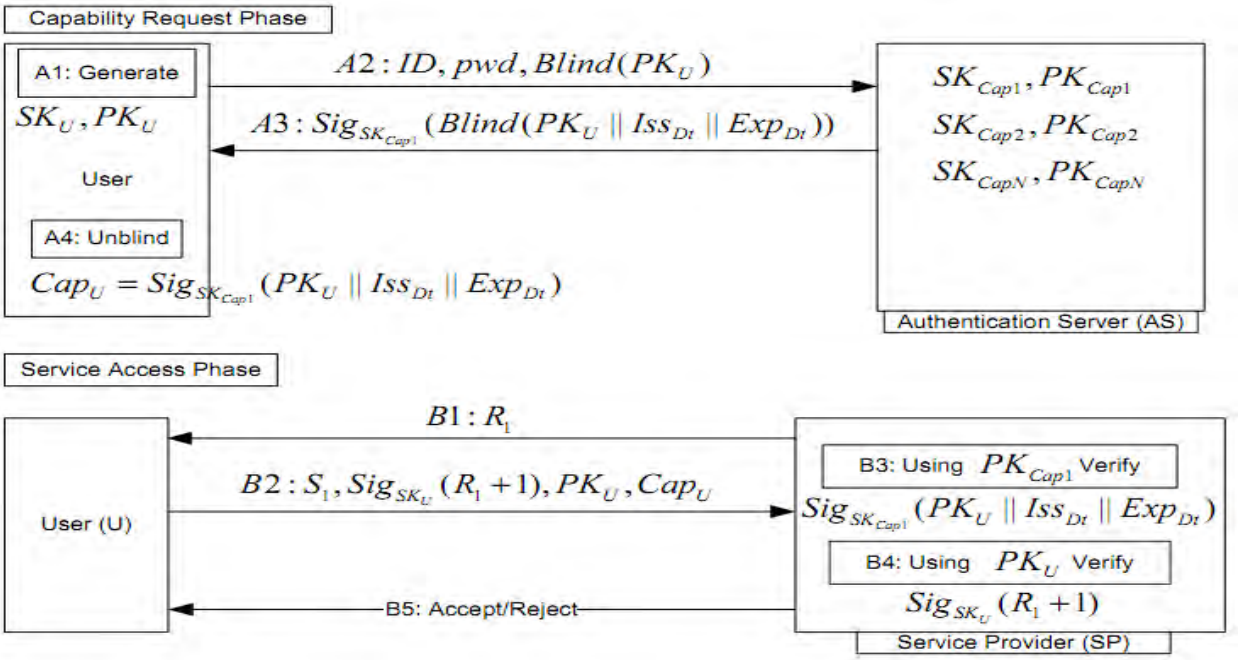


Figure 3-4: *System Architecture*

3.9 Avoiding Privacy Violation for Resource Sharing in Ad hoc Networks of Pervasive Computing Environment

The availability of personal information in pervasive computing environment may lead to the violation of privacy during resource sharing. The paper presents a model for privacy aware resource sharing. In this paper privacy is analyzed in terms of resource sharing by assigning policies to each resource in order to share resources among handheld devices without worrying about privacy. The model is flexible as it works with default policies as well as policies defined by its users. Privacy Violation Avoider (PriVA) model is a privacy aware one to protect information disclosure during information sharing in the pervasive computing environment. This model enables its users to tag the resources as sharable or non-sharable. It provides default policies to each resource which can be modified by the users. PriVA has TagR module to tag resources available and ClsR module to categorize all resources into groups [32]. Figure 3-5 shows architecture of the system [32]. The privacy management techniques used in this architecture of the system are:

- Access rights management
- Access policy management
- Classification of resources
- Granularity awareness
- Constraints and permissions
- Information disclosure protection

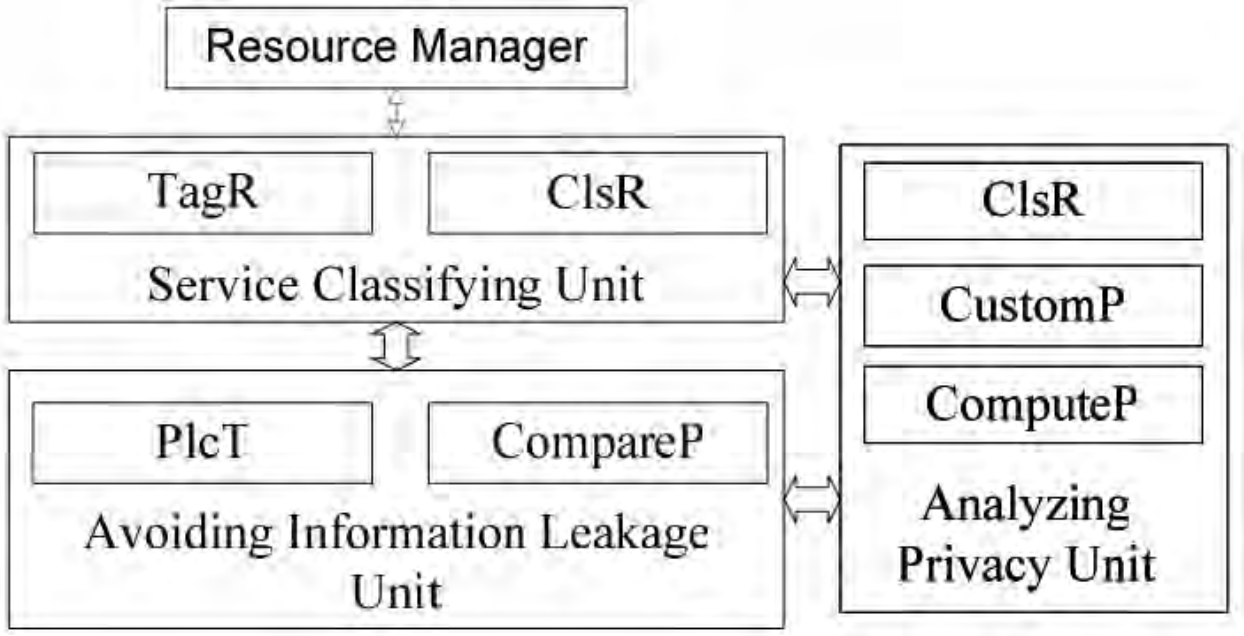


Figure 3-5: *Architecture of PriVA*

3.10 A Privacy Awareness System for Ubiquitous Computing Environments

Dealing with personal privacy is going to be a primary concern for the deployment of pervasive computing systems. This paper presents a privacy awareness system (PawS) for PCE which allows data collectors to both announce and implement data usage policies as well as providing data with technical means to keep track of their personal information as it is stored, used and probably removed from the system [31]. Any user in PawS can have privacy policy such as who can only view versus who can update or delete the data. When users enter any environment a privacy proxy checks these policies against the predefined privacy preferences. It encodes the privacy policies into XML format in order to classify the users' data. It also provides lifetime attribute for data in order to be controlled by the owners of the data. It allows its users to adjust the granularity of their location according to their need and provide to the system. User of PawS can also have the capability to assign ownership to their context data. The techniques mentioned above are some of the privacy management techniques PawS use to protect personal privacy of its users. Figure 3-6 shows overview architecture of the system [31]. The privacy management techniques used in this architecture of the system are:

- Access rights management
- Access policy management

- Classification of resources
- Data persistence control
- Granularity awareness
- Ownership of context
- Obscured information flow
- Information disclosure protection
- Protection of information usage

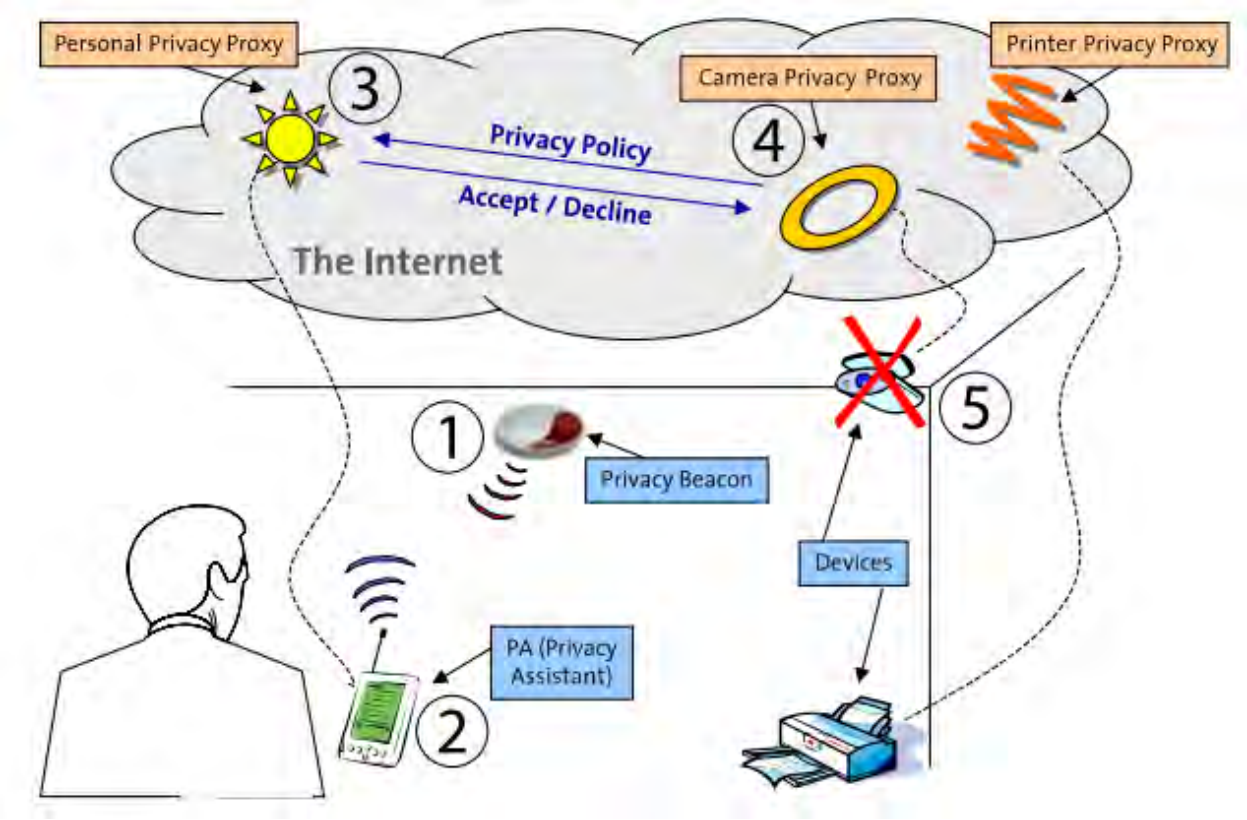


Figure 3-6: *Overview of the privacy management system*

Upon entering a ubiquitous computing environment with a number of data collections taking place (3, 4), optional services can be configured to suit the user's privacy preferences (5). Mandatory data collections (e.g., security cameras) can at least be detected (1) and collection details be recorded (2), allowing users or consumer interest groups to hold data collectors accountable for their statements.

3.11 Summary

From the related works we have understood that protecting security and privacy of users in PCE is difficult. The reason for this is that there is no unique security solution to every pervasive system since security issues of different systems may not be the same. As mentioned above, the different pervasive systems use different combination of privacy management techniques to overcome their specific security and privacy problems. Since a selected number of techniques used for a particular pervasive system may not be applicable for others, different combination of privacy protection techniques are used for the different pervasive systems. As we have understood from the review made, a particular technique may be used in a number of pervasive systems if there is a need. However, the approach of a particular technique may be different when applied on different pervasive systems based on the security requirements. The other important gap we have found is that the aim of the privacy management techniques used on the different pervasive systems is in order to balance the privacy concern of the users of the pervasive system due to the security hole of pervasive environments and the amount of information required by the pervasive system from the users. Pervasive systems need more information from the users of the system in order to improve efficiency with in pervasive system. However, it could be a potential threat if the huge amount of information collected, processed and stored in the PCE is accessed by unauthorized user in the pervasive environment. But in pervasive healthcare environments there is an additional problem beyond the security hole of the pervasive environments. There are medical information end users who may not have a direct communication with the pervasive healthcare of individual patients. The end users of medical information get the required information from the healthcare givers. In such cases healthcare givers may not get the owner's consent easily before any medical information to the secondary end users and a way that is helpful to their need. This problem also will be another focus of this thesis work. Table 3-1 shows the names of the pervasive systems explained in this chapter and other PCEs with their combination of techniques used [48].

Table 3-1: *Summary of privacy management techniques in PCEs*

Technique / Architecture	Access rights management	Access policy management	Classification of resources	Data Persistence Control	Granuality awareness	Constraints and permissions	Ownership of Context	Obscured information flow	Service access protection	Information disclosure protection	Protection of information usage
Capability-Based Privacy-Preserving Scheme	✓		✓	✓	✓	✓		✓	✓		
Context Models for Privacy	✓		✓		✓	✓	✓			✓	
Daidalos	✓	✓	✓						✓	✓	
Geopriv								✓	✓	✓	
Hierarchical Identity-Based Encryption	✓		✓	✓	✓	✓	✓	✓		✓	
Information Space Model					✓	✓		✓		✓	
Interaction History								✓	✓	✓	
LocServ								✓	✓	✓	
Mist								✓	✓	✓	✓
Mixed Networks								✓	✓		
MOSQUITO	✓	✓				✓				✓	
PawS	✓	✓	✓	✓	✓		✓	✓		✓	✓
PerGym/CARE	✓	✓			✓			✓			
PriVA	✓	✓	✓		✓	✓				✓	
Privacy Protecting Middleware	✓	✓					✓	✓	✓	✓	✓
Pseudonyms and Mix Zones								✓	✓	✓	
PSIUM	✓							✓	✓	✓	✓
Publish/Subscribe Substrate	✓	✓			✓	✓			✓		
Quality of Privacy (QoP)	✓					✓		✓			✓
RAVE	✓	✓				✓	✓				✓
SPARCLE	✓	✓			✓						
Trusted Platform Module	✓		✓				✓	✓	✓	✓	✓

Chapter 4 : Study on Privacy Related Concerns at the Black Lion Hospital

Though privacy concern of patients is worldwide issue, it is important to assess the issue in a particular hospital in order to understand the nature of the concern in Ethiopian context and solve the problem. In the first subsection, we will present background of the Black Lion Hospital (Tikur Anbesa in Amharic). In the second subsection, we briefly discuss on the summary result of the assessment made based on the questionnaire. In the third subsection, we will present the result of the interview made with HIV patients. Finally, the lessons learnt will be presented.

4.1 Background of the Study

Black Lion Hospital is located in the capital of Ethiopia called Addis Ababa. Black Lion Hospital is the largest referral public hospital in the country. It is handed to Addis Ababa University (AAU) in 1998 E.C by the ministry of health (MOH) as a main teaching hospital. Medical faculty of the university was hosted in the hospital compound long before the hospital was handed to the university. The medical faculty (now renamed as the College of Health Sciences) is staffed with senior specialists and is the largest and the oldest health training institution in the country. It is administered by AAU and teaches about 300 medical students and 350 staff of residents every year. It is 24 hours open for emergency services and provides a tertiary level referral treatment. The hospital has 800 beds, 130 specialists and 50 non-teaching doctors. It offers diagnosis and treatment for approximately 370,000-400,000 patients each year. The hospital sees around 80,000 patients every year [67].

The thesis work includes information gathering and analysis on privacy concern of patients for the case of Black Lion Hospital. A questionnaire consisting of 19 questions is prepared and distributed in the departments of Internal Medicine, surgery, pedantrics and Gynecologist/ Obstetrician (Gyn/Obs). The questionnaire includes 17 closed ended and 2 open ended questions (Appendix A). A total of 50 questionnaires are returned out of the 90 distributed questionnaires. The questionnaires were disseminated for physicians and nurses. From the 50 questionnaires 44 of them are from physicians and 6 of them are from nurses. An interview is also made with a total of 63 HIV patients (inpatients and outpatients) on the issue of privacy concern when their medical information is shared for different purposes. The size of the sample we took for the

study is relatively small due to availability issue. However, a number of surveys are already conducted on patients' privacy concern as stated by Appari and Johnson [49] and our survey is made as a plus on the internationally known privacy issues of patients. The tool used for the data analysis is SPSS. The result compiled from the questionnaire and the interview will be presented in the subsequent subsections.

4.2 Healthcare Providers View on Privacy Concerns

The assessment results of the most important questions are presented graphically and the rest presented in table form. In most of the cases a respondent answers more than one option for a given question which leads to the sum (%) for the options exceed 100%.

From the health providers' point of view, the patients that are most worried about their privacy protection are HIV patients. 92% of them believe that HIV patients are most worried by privacy, those answered as Psychiatric behavior is 38% , those answered as Sexually transmitted infections is 38% and who answered other is 2%.The statistical analysis result of this data is graphically represented as shown in Figure 4-1.

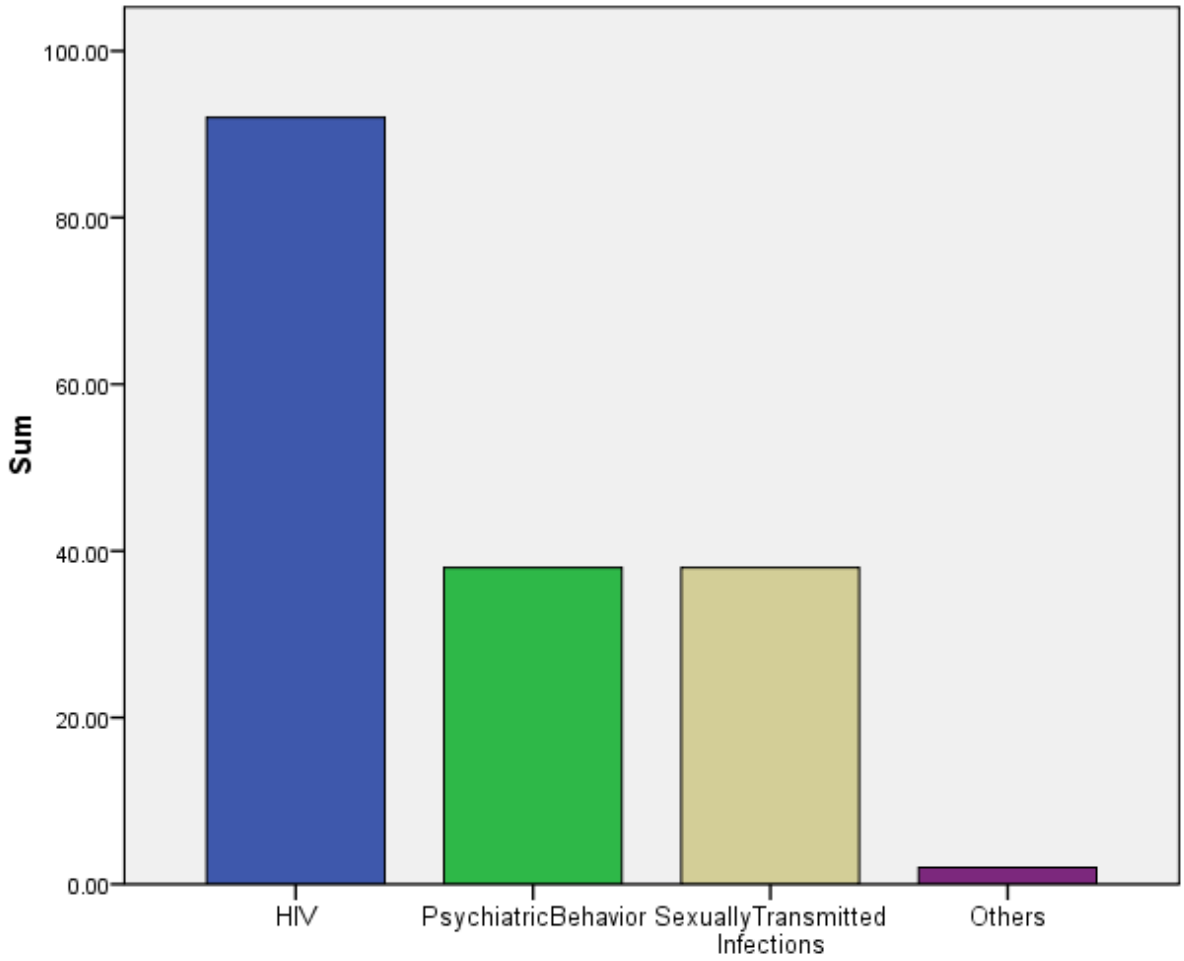


Figure 4-1: *Patients privacy concern on diseases*

Regarding the sources for privacy concern of patients, 88% of them believe that it is because of shame and disgrace, 80% of them believe that it is due to fear of social discrimination, 8% of them trust that it is because of the legal implications it has, 4% of them believe that it is due to insurance issues and 2% of them answered other. The statistical analysis result of this data is graphically represented as shown in Figure 4-2.

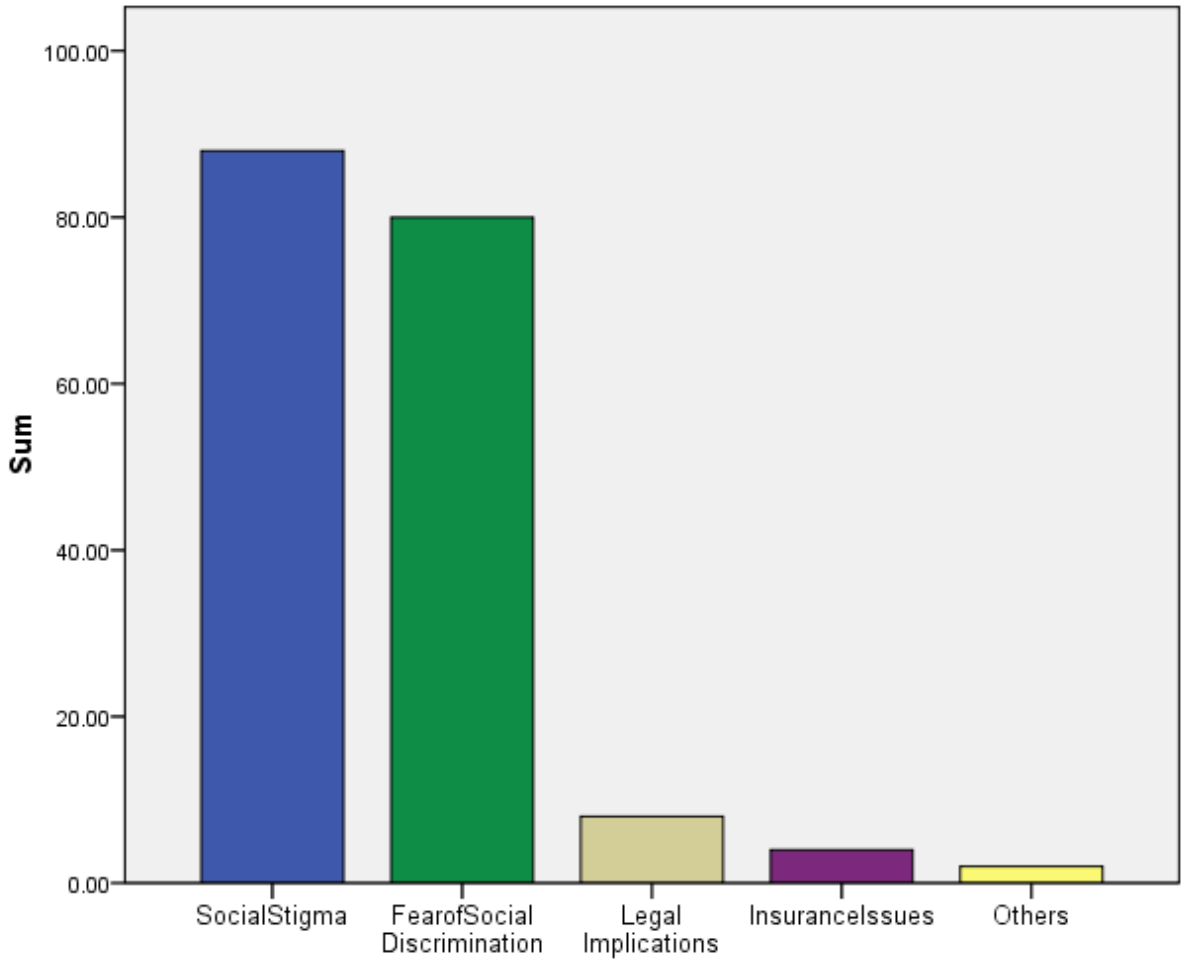


Figure 4-2: *Sources of privacy concerns of patients*

To store information of the most privacy concerned patients, 92% of them use Manual system and only 8% of them use computer based system. The statistical analysis result of this data is graphically represented as shown in Figure 4-3. Those of the computer system users believe that it minimizes risk because of the access limitation for others.

Those who use manual system to store the data uses the following techniques:

- Keeping the patients' card securely (in locked cabinet)
- Keeping patients privacy using unique numbers
- Writing patients privacy concern on their charts
- Coding (using words which are easy for physicians only, Abbreviations of medical terms)
- Avoiding passing charts to patients or attendants

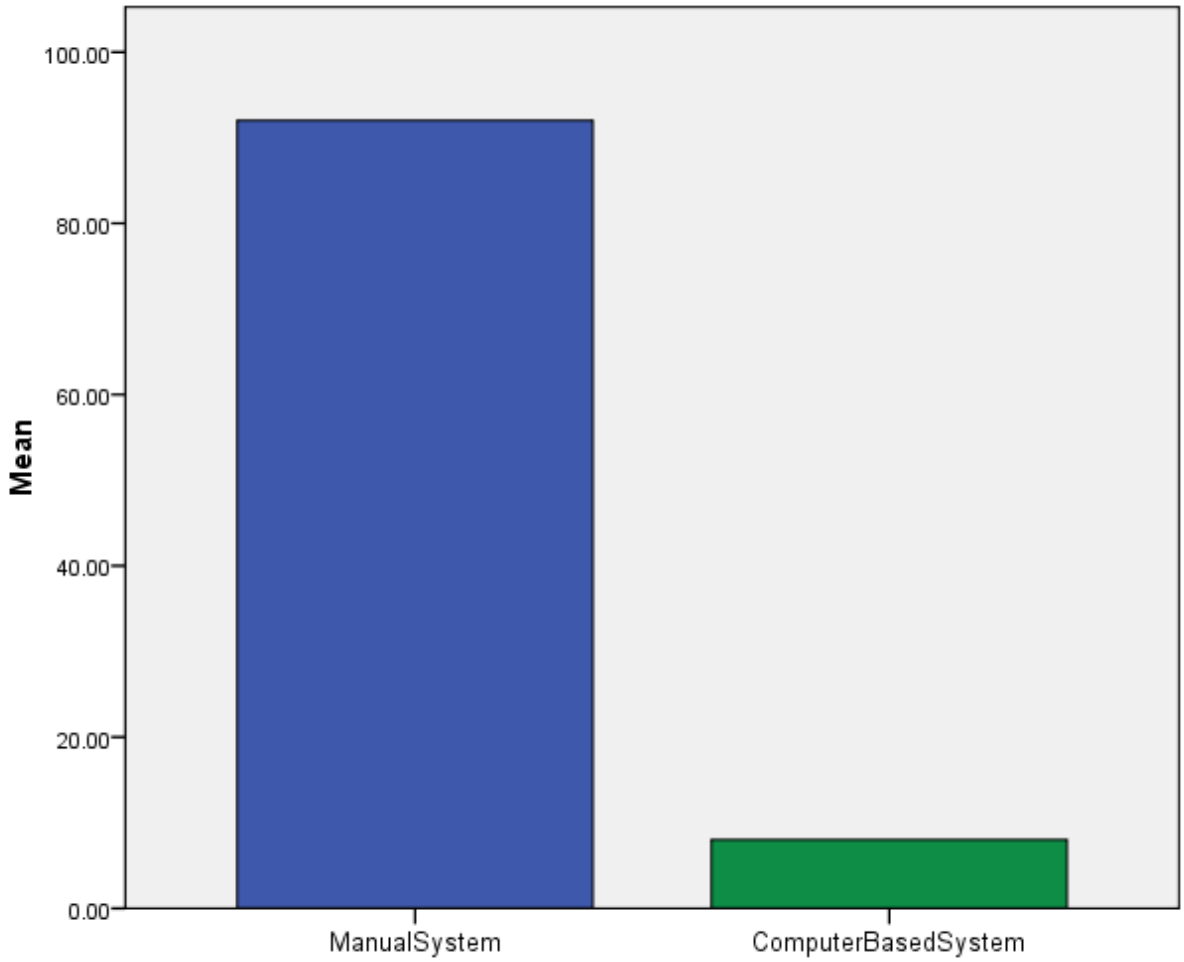


Figure 4-3: *Systems used to keep information of most privacy concerned patients*

Handling patients' records manually can be source of privacy concern to patients. According to the survey, 62% of them believe that patients' concern is due to Un-authorized access made on the records, 78% of them believe that their concern is due to fear of poor handling or storage, 22% of them believe that it is because of disasters can happen and only 2% of them answered other. The statistical analysis result of this data is graphically represented as shown in Figure 4-4.

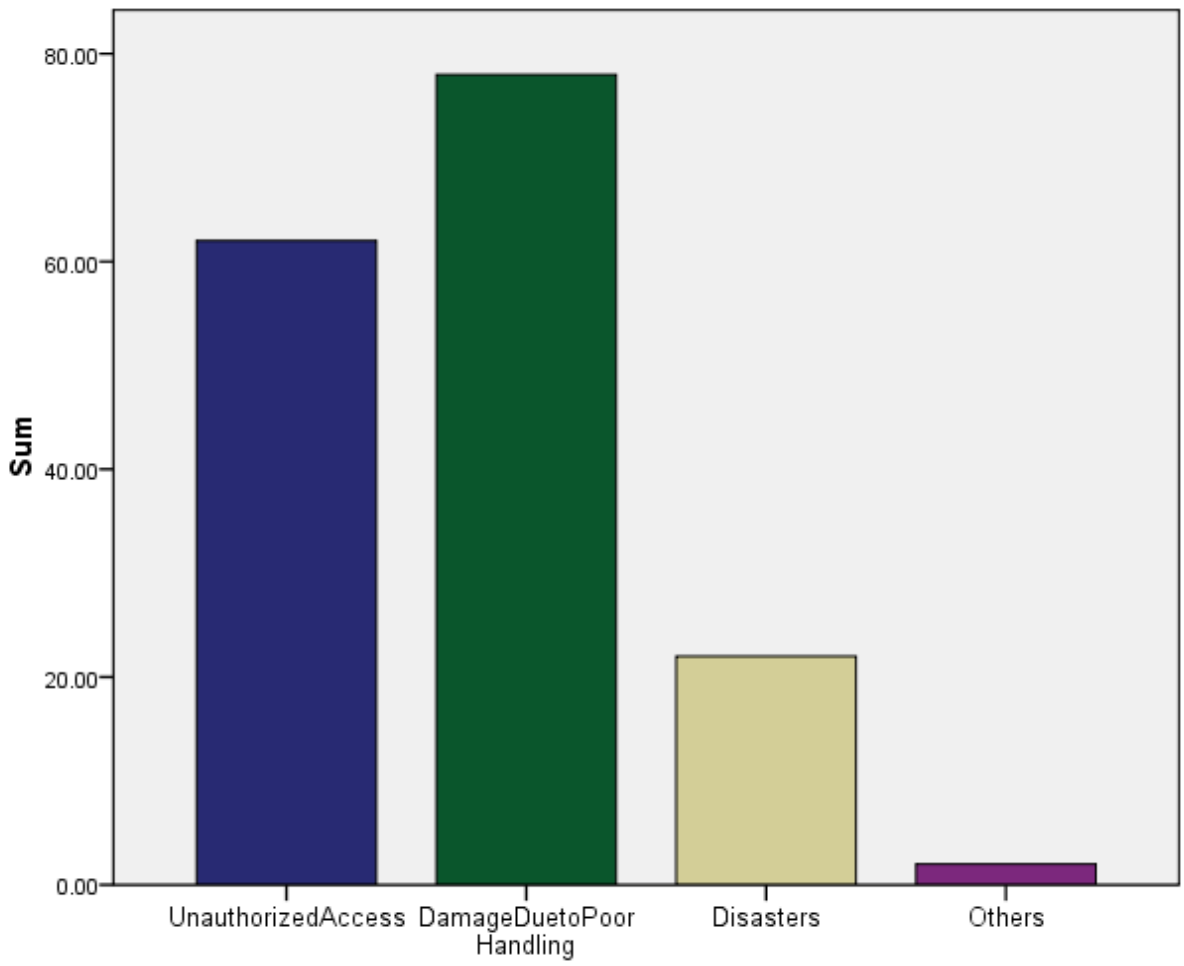


Figure 4-4: *Fear of patients on manual record system*

The survey also indicates that privacy concern of HIV patients varies among the different age rage. 10% of them answered as < 15, 52% of them as 15 – 24, 70% of them as 25 – 34, 28% of them as 35- 44, 34% of them as 45 – 54, 26% of them as 55 – 64, 22% of them as 65 – 74 and 14% of them as >74. The statistical analysis result of this data is graphically represented as shown in Figure 4-5.

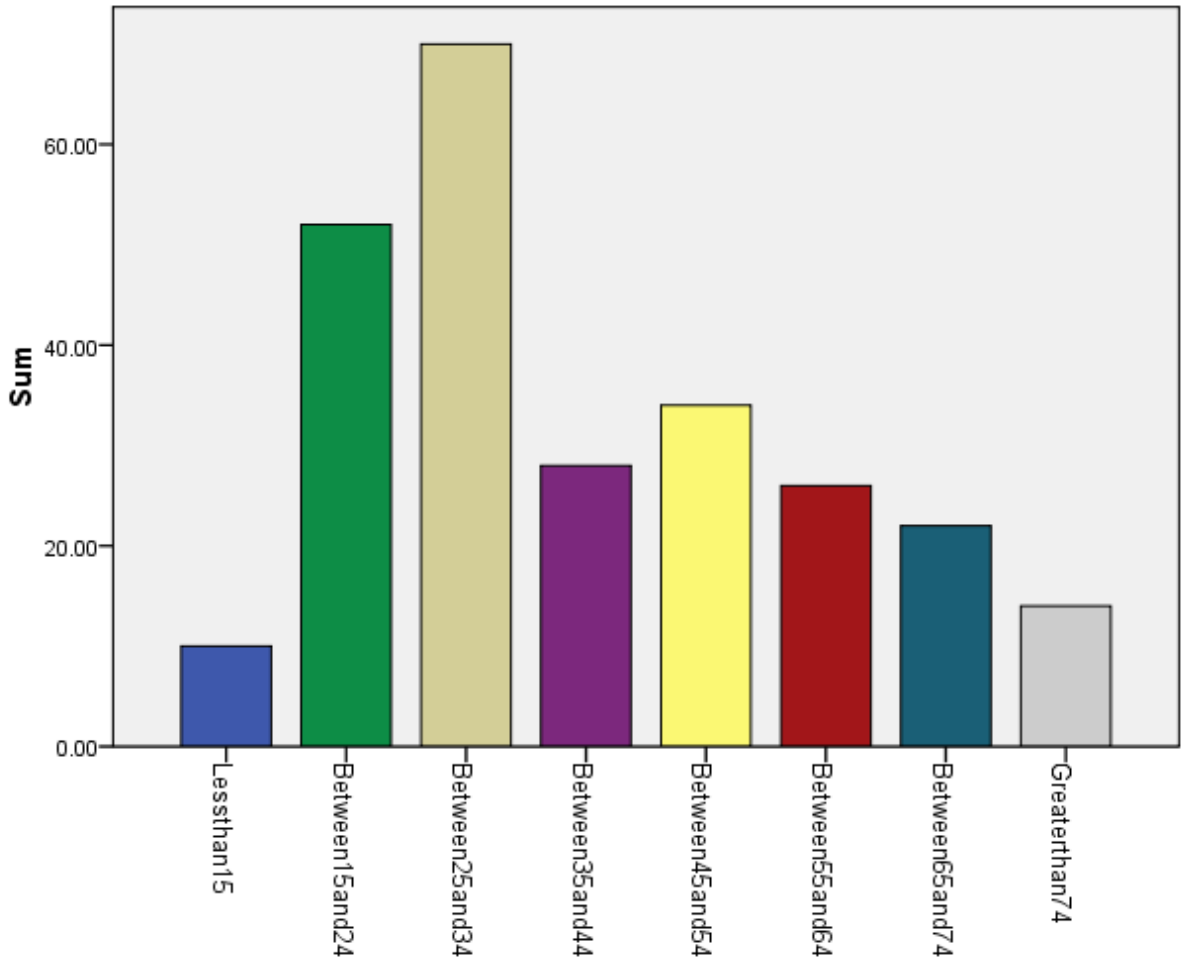


Figure 4-5: *Impact of age on the privacy concern of patients*

Result of the survey indicates that healthcare providers use different approaches while sharing medical information with other users of medical records in order to protect privacy of the owners of the information. Only 26% of them provide information by removing all patient specific identifiers, only 8% of them use reporting approach to the responsible body, only 68% of them use coding/encryption techniques to protect privacy and 40% of them ask patients' consent before providing their information to others. The statistical analysis result of this data is graphically represented as shown in Figure 4-6.

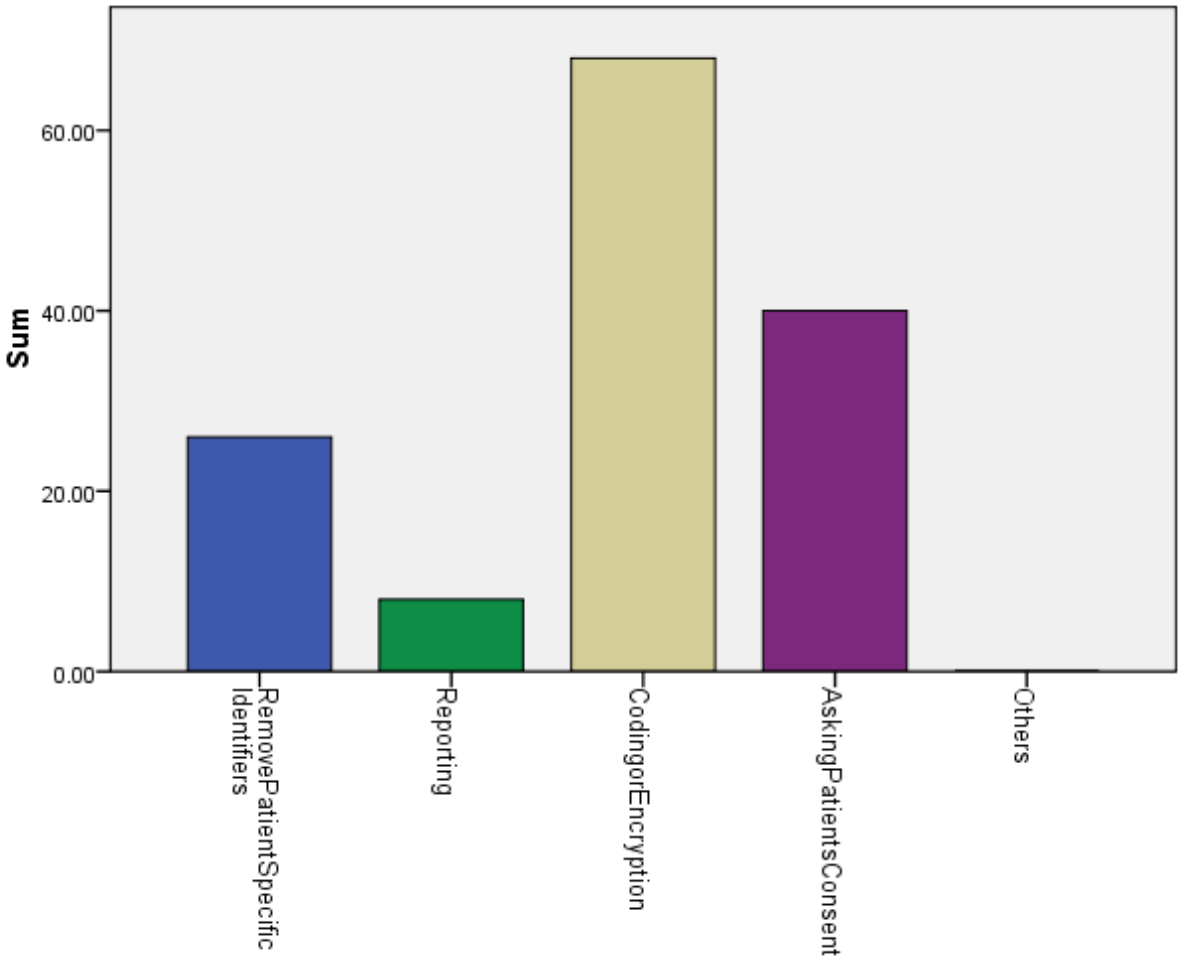


Figure 4-6: *Privacy protection approaches*

The result of the survey also indicates that healthcare providers have similar view in the impact of privacy violation of patients. 72% of them believe that it creates psychological distress, 82% of them have fear of patients' loss of trust in healthcare team and system, 68% of them believe that it may lead to reduced adherence to treatment and care and only 2% of them believe other. The statistical analysis result of this data is graphically represented as shown in Figure 4-7.

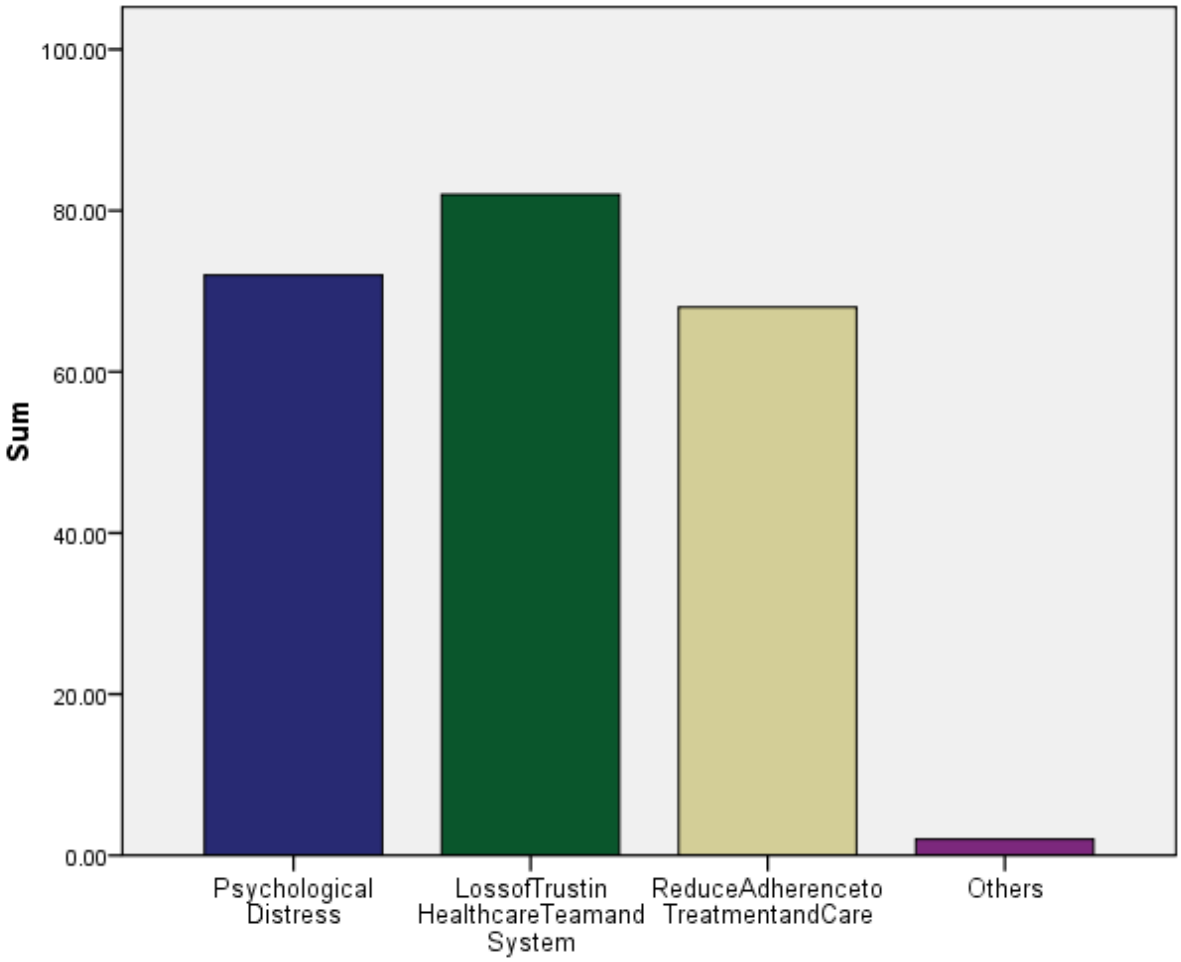


Figure 4-7: *Impacts of privacy violation of patients*

The assessment results of the questions presented in table form are as follows:

Most of the healthcare providers believe that digitized systems can improve privacy of the owners. The statistical analysis result is presented as shown in Table 4-1.

Table 4-1: *Impact of digitized health record systems on improving privacy of patients*

Yes	No	May be	No comment
62 %	4 %	32 %	2 %

Most of them also believe that there are cases that force them to provide medical information of patients to other users of medical records. The statistical analysis result is presented as shown in Table 4-2.

Table 4-2: *Sharing medical information with others*

Yes	No
76 %	24 %

Some of the cases mentioned by the healthcare providers are:

- if the failure to disclosure puts third party at risk
- family or friends eager to know their colleagues' result
- medical consultation
- for children under 9 years old
- if unconscious and to decide management
- to give better medical care with the help of other healthcare professionals (senior experts)
- if it puts health of staff team at risk and mission violence
- if medical information is required by law or ordered by court
- for teaching purpose (among physicians and practitioners)
- issues related to job position and mental conditions

Most of them believe that some of the patients only refuse to reveal important information to physicians due to fear of information disclosure. The statistical analysis result is presented as shown in Table 4-3.

Table 4-3: *Patients fear to disclose their information to their physicians*

All of them	Most of them	Some of them	None
4 %	16 %	70 %	10 %

Most of them also believe that there is no system that allows patients by which they can approve sharing of their medical information with others. The statistical analysis result is presented as shown in Table 4-4.

Table 4-4: *A system for patients to approve their medical information for access*

Yes	No
46 %	54 %

Most of them believe that some of the patients only agree on sharing their medical information among physicians and reject the notion of releasing to third parties including employers and family members. The statistical analysis result is presented as shown in Table 4-5.

Table 4-5: *Patients those agree on sharing their medical information with other physicians*

All of them	Most of them	Some of them	None
2 %	38 %	56 %	4 %

Most of them also believe that controlling their patients remotely via their smart phones is more jobs for physicians. The statistical analysis result is presented as shown in Table 4-6.

Table 4-6: *The impact of monitoring patients remotely*

It simplifies physicians' job	It over exposes patient information	It is more jobs for physicians	Others
38 %	40 %	42 %	6 %

Most of them believe that pervasive systems should be secure. The statistical analysis result is presented as shown in Table 4-7.

Table 4-7: *Security level of pervasive systems from healthcare providers' point of view*

Highly secure	Secure	Less secure	Unclassified
40 %	48 %	10 %	2 %

Most of them are unfamiliar with the security and privacy policy of the country's health system. The statistical analysis result is presented as shown in Table 4-8.

Table 4-8: *Familiarity with the security and privacy policies of the country health system*

Very familiar	Familiar	Less familiar	Unfamiliar
4 %	28 %	30 %	38 %

Most of them believe that most of the time they respect medical rules while treating patients in the hospital. The statistical analysis result is presented as shown in Table 4-9.

Table 4-9: *Healthcare providers respect medical rules*

Always	Most of the time	Sometimes	Never
38 %	48 %	12 %	2 %

Most of them are unaware about the Ethiopian policy on the protection of data, information and devices used to process and store them. The statistical analysis result is presented as shown in Table 4-10.

Table 4-10: *Healthcare providers' awareness on Ethiopian data protection policy*

Extremely aware	Very aware	Moderately aware	Slightly aware	Unaware
6 %	10 %	12 %	22 %	50 %

The survey result of the two open ended questions is presented as follows:

Healthcare providers may use different privacy protection mechanisms in their existing system. The survey indicates that only 4% of them use computer based privacy protection mechanism for their patients' medical information and 88% of them protect privacy of their patients in their manual system by doing the following practices:

- Keeping patients' file in the hospital at legally protected place (archive)
- By discussing the case of privacy with the patients
- Using abbreviations (short forms)
- Using words that physicians only can understand (code)

Sharing medical information among healthcare providers and other users of medical information is a common practice in the dominating existing system (manual system). The survey indicates that healthcare providers have different believes about the importance of sharing information in the digitized world. Only 44% of them have positive responses and 18% of them have negative responses.

Some of the positive responses were:

- it is important
- it will be reliable, easy to access and trusted
- sharing will be easier
- it facilitates a better privacy
- it is fast, reliable and simple
- it will be ideal if done with backup

Some of the negative responses were:

- it makes patient privacy concern at risk. So that, some protection mechanisms are needed before sharing medical information
- sharing for concerning bodies involved in patient management directly or indirectly is critical for patients
- within the limit of good practice and conserved ethics, in accepted sharing of documents

4.3 Patients View on Privacy Concerns

The most commonly known diseases that make patients fear of social stigma and discrimination are HIV and psychiatric behavior [49]. An interview is made with 35 outpatients and 28 inpatients of HIV patients. The interview was about privacy concern of the patients when their medical information is disclosed due to different reasons. Their response of the inpatients and outpatients is organized in table form separately.

The survey result indicates that 89.28% of the HIV inpatients do not worry about their information disclosure. The result is presented as shown in Table 4-11. One of the reasons mentioned by them is that they are feeling "hopelessness" and that is why they do not care about the social stigma and discrimination.

Table 4-11: *Inpatients view on privacy concerns*

Consent is required	Provide without any consent	Unwilling to disclose for any case
7.14 %	89.28 %	3.57 %

The survey result also indicates that 71.42% of the HIV outpatients do not allow their medical information to be accessed without their consent. The result is presented as shown in Table 4-12.

Table 4-12: *Outpatients view on privacy concerns*

Consent is required	Provide without any consent	Unwilling to disclose for any case
71.42 %	17.14 %	11.42 %

4.4 Findings and Lessons Learnt

From the findings we can have some decision that helps us to accomplish our proposed thesis work successfully.

The findings are:

- Outpatients are more concerned about their privacy than the inpatients since social stigma is the main challenge for them.
- Some of HIV outpatients also worry about transportation cost to go to the BLH to check their CD4 count and take treatment periodically.
- Most of the patients worry about their medical records specially HIV carriers.
- In the case of BLH, health records are managed manually.
- Physicians use coding/encryption in order to hind identity of individuals before providing medical information to others.
- Privacy concern of patients is more visible at the age range of 15-24 and 25-34.
- The mainly known impacts of privacy violation on patients are: psychological distress, loss of trust in healthcare team and system, and reduce adherence to treatment and care.
- There are many situations of sharing medical information and there is no a desired system that helps patients fully control their medical information.
- Some patients may refuse to reveal their information to their physicians.
- Most of the healthcare providers believe that monitoring patients remotely using pervasive systems is not good because of privacy reasons.
- Most of healthcare providers are unaware of Ethiopian policy on protection of data.
- They use privacy protection techniques such as medical terms (abbreviations), chart numbers and so on.

The lessons learnt are:

- Privacy concern of individuals depends on the level of consensus of the people since most of HIV patients believe that this time social stigma is relatively less.
- There are patients that do not want to reveal their information to their physicians and it is true that patients have a full right to do so.

- The privacy protection techniques used by physicians on the existing manual system may not be according to the patients need. It is also true that if the request is on HIV medical information or other sensitive medical information, physicians use a general conclusion on providing the information by using codes as the patients' name or disease name. However, in some cases such information may not be helpful for the end users.
- HIV patients go to the hospitals periodically to check their CD4 count and/or take their treatment. However, sometimes patients may go back home without taking the medicine when they get someone around who does not know about their case. This indicates that how much the patient have a deep feeling on privacy concern.

Chapter 5 : The Proposed System

The proposed system is about the development of Privacy Aware Pervasive Healthcare System (PAPHS). The system is applicable for HIV patients. This is because HIV patients are more concerned about privacy than other patients. Patients may refuse to divulge important information in cases of health problems such as psychiatric behavior and HIV as their disclosure may lead to social stigma and discrimination [49].

Though focus of this thesis work is on managing personal privacy of patients in the PCE, we are forced to come through a pervasive healthcare system in order to implement our work. The system deals with active monitoring of the human immune system in both HIV patients and individuals who are regarded as at-risk and is important to decide when a particular patient has to start ART treatment and other related drugs. The most common and reliable technique to monitor human immune system is CD4 (T-cell) count in the patient blood. The current and imminent devices are enabled by microfluidic solutions [68]. At the end of this thesis work we are going to come up with the privacy aware pervasive healthcare system. In this system the amount of patients' medical information required by the pervasive system in order to satisfy the needs of users of medical information and the personal privacy of the patients will be balanced as a result of the privacy manager module that will be integrated with the pervasive healthcare system.

5.1 Architecture of the PAPHS

The proposed Privacy Aware Pervasive Healthcare System (PAPHS) in this thesis work includes four main modules as shown in Figure 5-1. These are: the patient module, the patient privacy protection manager module, the healthcare module and the physician module. The figure shows architecture of the PAPHS. In subsequent subsections, each module of the system will be explained incrementally and in detail.

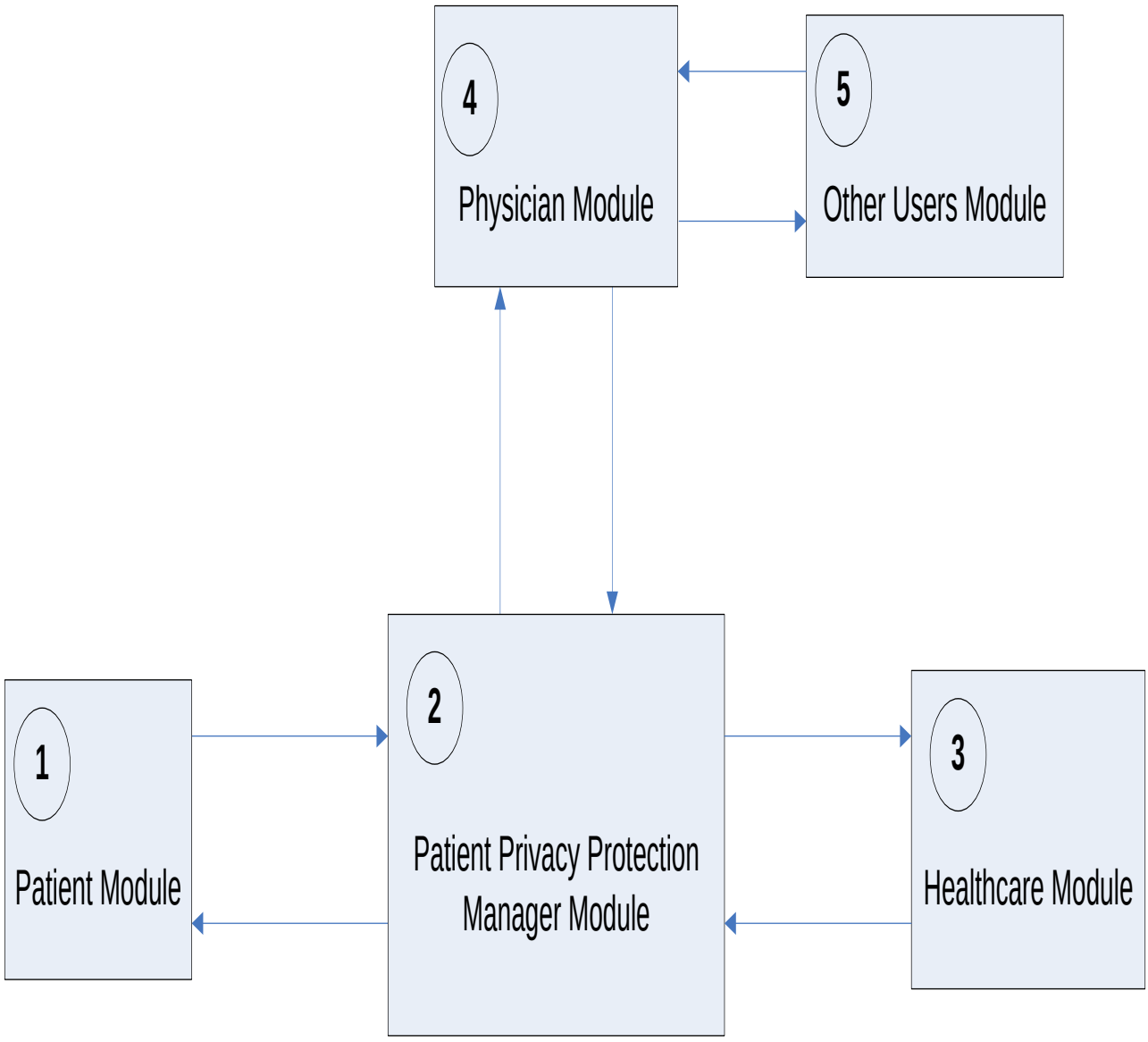


Figure 5-1: *Architecture of the PAPHS*

5.1.1 The Patient Module

This module contains a number of components that resides on the patient terminal. The module includes the Moxi z automated cell counter and the patient's smartphone. The cell counter has the ability to count CD4 cell or T-cell of an HIV patient and pass the reading to the smartphone of the patient via Bluetooth. Then the smartphone sends a message (SMS) based on the current reading to the respective physician's smartphone via web based services if and only if that reading is important for decision making. Figure 5-2 shows the Moxi z automated cell counter

device. Figure 5-3 shows the detail architecture of this module. In this subsection each of the components will be explained in detail.

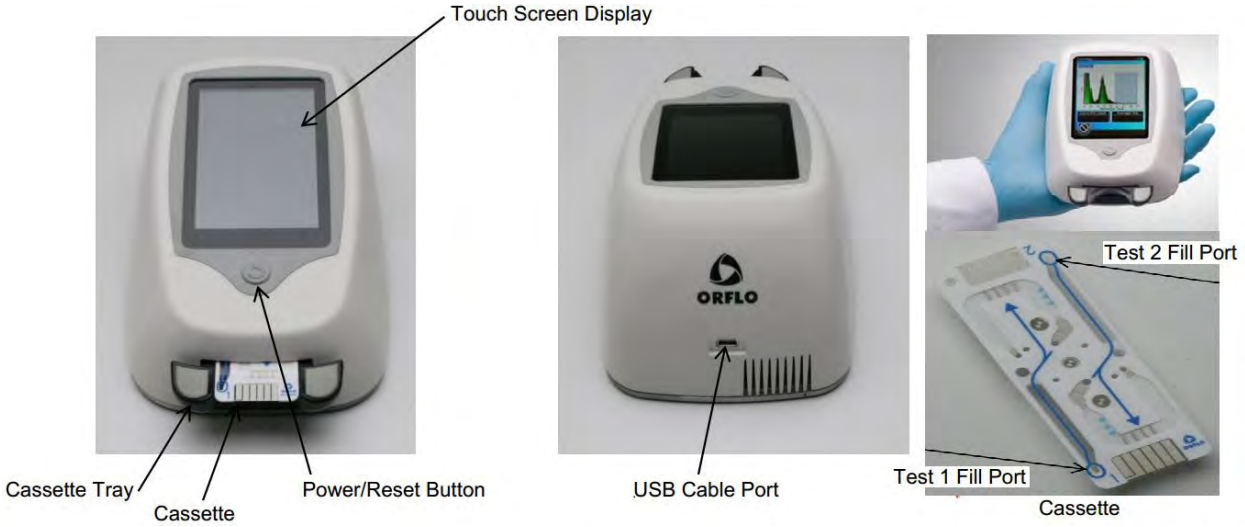


Figure 5-2: *Moxi z Mini automated cell counter kit*

(Source: http://www.orflo.com/Moxi%20Z%20User%20Manual%20rev%204-1.pdf)

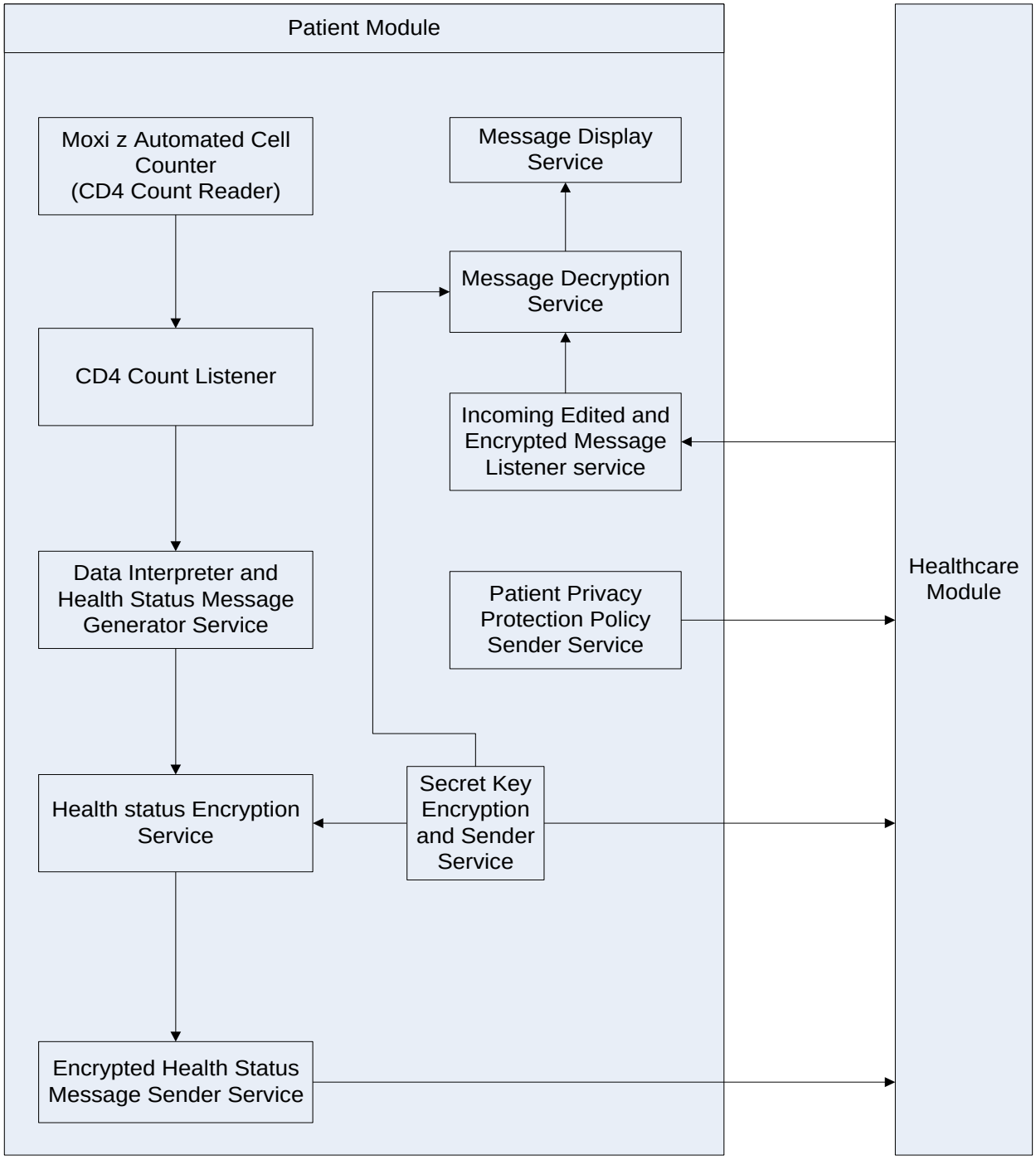


Figure 5-3: *Architecture of the patient module*

5.1.1.1 Moxi z Automated Cell Counter Component

Moxi z mini automated cell counter is the only cell counter which combines Coulter principle that is usually used in high end cell counters with a patented thin film sensor technology in order to allow for highly accurate (> 95%) cell count. At the same time today's workflows need

accurate quality of samples, determining precise cell count has significant effect on the outcomes and costs.

Moxi z is a hand held Bluetooth enabled device and Produce repeatable cell counts in just 8 seconds. It is ideal for a millions of cell counts and it can be applied to a wide variety of cell counting applications because of its broad dynamic range. Such applications include white blood cell counts, red blood cell counts, etc. This ultra-small device uses patented microfluidic thin-film cassettes that enable automatic load and measure tasks. Moxi z is completely technique independent and provides the ideal alternative to the tedious manual counting in cytometers, or the inaccurate results in image based automated cell counters with accuracies between 75 and 80 percent [69]. In this thesis work Moxi z is used as a sensor of CD4 count for HIV patients.

CD4 cells are a type of white blood cells and are sometimes called T-cells. They send signals to activate human body's immune response when they detect intruders such as virus or bacteria. There are millions of CD4 cells families and each family is responsible to fight a particular type of germ. When HIV infects humans, most of the time it infects the CD4 cells, finally they become part of the cells. When the number of CD4 cells increase to fight an infection, they also make more copies of HIV. If someone is infected with HIV for a long period of time, the number of his/her CD4 count gets decrease. This shows that the immune system is being weakened and the person more likely will get sick. CD4 cells are usually reported as the number of cells per cubic millimeter (cells/mm3). The normal CD4 cell counts are between 500 and 1600. If the CD4 count goes below 350 cells/mm3 most healthcare providers initiate ART for those patients. According to the previous WHO recommendation, set in 2010, was to offer treatment at less than or equal to 350 CD4 cells/mm^3. 90% of all countries have adopted the 2010 recommendation [70]. In some cases CD4 cell count of HIV patients may drop to zero. If someone has a CD4 count less than 200 cells/mm3, it is one of the qualifications for a diagnosis of AIDS. It is also important to know when to start certain types of drug therapy and together with viral load in order to estimate for how long someone can stay healthy. Figure 5-4 shows the cassettes used for counting number of cells with in a blood solution and the screen of the Moxi z where the Bluetooth icon is found. Based on the sample of a blood solution on the cassette, the count is displayed on the screen as shown in Figure 5-5.

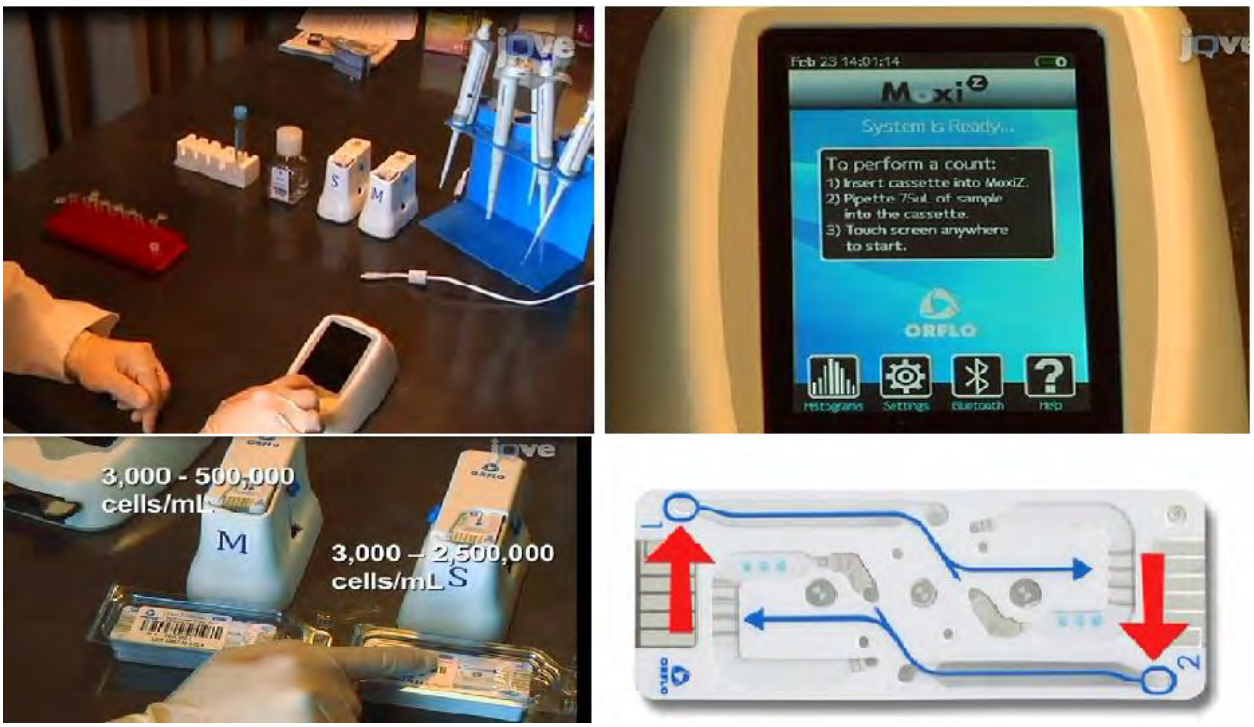


Figure 5-4: *Moxi z Mini automated cell counter accessories*

(Source:http://www.jove.com/video/3842/determination-mammalian-cell-counts-cell-size-cell-health-using-moxi) Retrieved on September 06, 2013 at 04:49:55

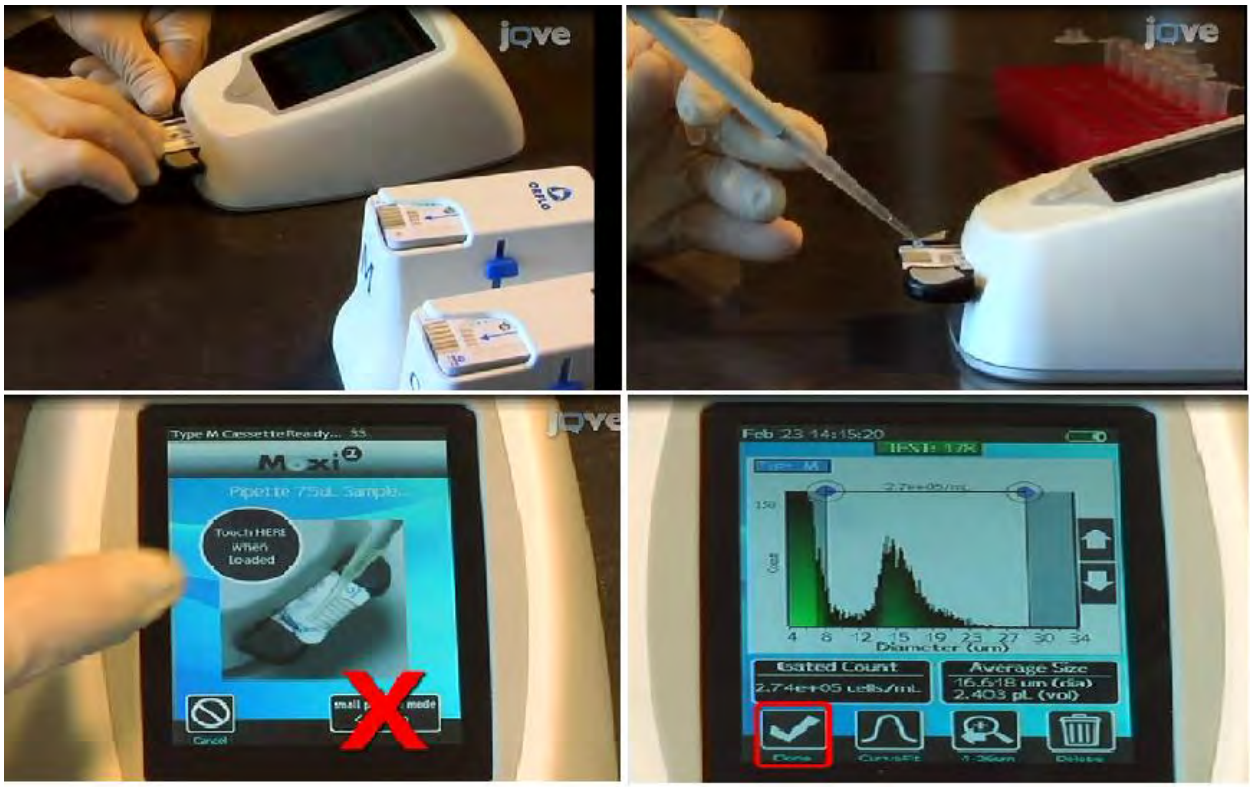


Figure 5-5: *Pipette the sample to start counting*

(Source:http://www.jove.com/video/3842/determination-mammalian-cell-counts-cell-size-cell-health-using-moxi) Retrieved on September 06, 2013 at 04:49:55

5.1.1.2 CD4 Count Listener Component

This is the main component in the patients' module. The role of this component is acquiring the CD4 reading on the Moxi z device via Bluetooth. Figure 5-6 shows Moxi z is a Bluetooth enabled device which can send the reading (data) to another Bluetooth enabled devices such as smartphones.

Figure 5-6: *Connecting Moxi z with a PC via Bluetooth*

(Source:http://www.jove.com/video/3842/determination-mammalian-cell-counts-cell-size-cell-health-using-moxi) Retrieved on September 06, 2013 at 04:49:55

5.1.1.3 Data Interpreter and Health Status Message Generator Service Component

The data interpreter gets CD4 count from the CD4 count listener in order to interpret and generate health status of the patient. Message is generated if the CD4 count of the patient is less than or equal to 350. The data interpreter identifies the CD4 count as if <=350, <300, and <200. To start with the interpreter, if the CD4 count is less than 200 it is one of the qualification for diagnosis of AIDS. It is also important to know which opportunistic infections can happen at this level. If the CD4 is less than 75 and the patient have the signs of fever, night sweets, abdominal pain, diarrhea and fatigue, the physician should also check for Mycobacterium avium complex [MAC]. If the CD4 is less than 100 and the patient has headache, fever, fatigue, neck stiffness and/or memory loss and mood change which are the signs of Toxoplasmosis and Cryptococcosis, so the physician should also treat the patient with anti-fungal medication. If the CD4 count is less than 200 the physician should understand that there is highest risk of developing Pneumocystis pneumonia [PCP]. If the CD4 count is less than 300, the physician should initiate ART and also initiate Strong Antiretroviral drugs (ARVs) to prevent Pneumocystis pneumonia [PCP] which

shows the first signs of fever, shortness of breath, chest pain and dry cough. If the CD4 count is less than or equal to 350, the physician should initiate ART for that patient. This is the way the CD4 count is interpreted and Message is generated accordingly in a way that helps the physicians. The message will not be shown directly to the patient rather it will be sent to the respective physician then the physician will edit the message and send it back to the patient.

5.1.1.4 Health Status Encryption Service Component

Once the message is generated it should be obscured using blowfish symmetric cryptosystem before leaving the patient mobile terminal. The message encrypted will be delivered to the next component to be sent to the corresponding physician.

5.1.1.5 Encrypted Health Status Message Sender Service Component

The encrypted health status message will be sent to the respective physician of the patient via web server. The message will be sent to the healthcare module then to the physician mobile terminal.

5.1.1.6 Incoming Edited and Encrypted Message Listener Service Component

The message edited by the physician is encrypted using blowfish technique and sent to the patient terminal as it is. The message edited by the physician of a particular patient is sent via the web and received at this component of the patient terminal.

5.1.1.7 Secret Key Encryption and Sender Service Component

The secret key is 64 bit length which is shared among a patient and the respective physician. It is an input to the blowfish encryption and decryption algorithms that are implemented at both the patients and physicians mobile terminals. The secret key of a particular patient must be shared with the respective physicians in order to encrypt and decrypt the messages during communication. The secret key should be obscured before shared with the respective physicians. Therefore, it should be encrypted using Elliptic Curve (ECC) asymmetric cryptosystem and stored in the database. The encrypted secret key will be sent to the healthcare unit via this component and stored in the database in order to be available for the respective physician.

5.1.1.8 Message Decryption Service Component

Once the edited message sent from physician is delivered on the listener it will be decrypted using blowfish cryptosystem. The original message will be delivered to the next component for display.

5.1.1.9 Message Display Service Component

This is the graphical user interface on which the messages come from the physician are displayed on the patient's mobile terminal.

5.1.1.10 Patient Privacy Protection Policy Sender Service Component

The patient is the policy make on his/her own medical information. The privacy protection policy will be explained in detail in the patient privacy protection manager module. The policy is set by the owner of the information and sent to the healthcare module in order to govern the information access request and provision processes.

5.1.2 The Patient Privacy Protection Manager Module

This subsection deals with the privacy protecting techniques which is the ultimate goal of this thesis work. This module deals with the security and privacy of patients during communication and sharing of medical information among the physicians, patients and other stakeholders. This component includes various techniques to protect personal privacy of patients. These are:

- Obscuring information flow
- Users access rights management
- Users access policy management
- Granularity awareness
- Constraints and permissions
- Data persistence control
- Classification of resources

The techniques are selected based on the security assessment made on the proposed pervasive healthcare system. Since the wireless communication established between the healthcare module (central server) and the mobile users (physicians and patients) can be accessed by intruders, the information flow should be obscured by encryption techniques. The end users of medical

information should access patients' information based on the access rights given by the owners of the information. The system grant or deny access rights for the medical information end users according to the policy set by the owners of medical information. Patients' information is grained down in order make the anonymity set by the patients be according to the level of their need. Owners of medical information may need to grant time constrained access rights in order to deny the access right once the time is over. The system contains detail information of patients that are used for message communication process only among patients and physicians. However, the data will be huge and not suitable to be accessed by mobile devices. In order to control such huge data, the unnecessary data will be deleted by keeping the important medical history of the patients separately. Patients are required to classify each attribute of their information as sharable/not sharable depending on the identity of the end users and the kind of resource (disease) during setting policies on information access. The detail architecture of this module is as shown in Figure 5-7. The components it consists of are explained in this subsection in detail. This architecture shows a group of privacy protection techniques each may physically resides on one or more of the other modules of the PAPHS. The replica of this architecture is available at Appendix X in more detail.

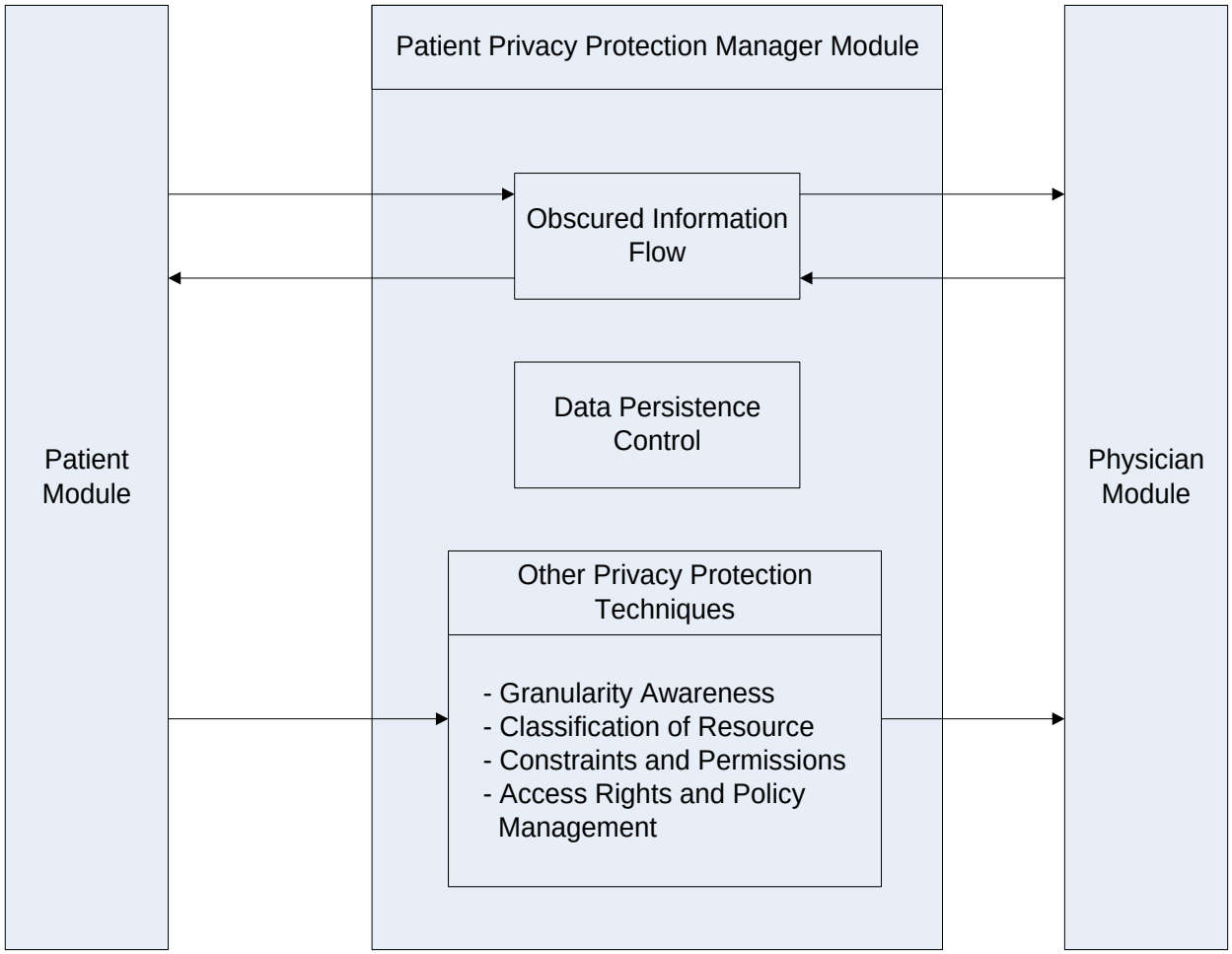


Figure 5-7: *Architecture of the patient privacy protection manager*

5.1.2.1 Obscured Information Flow Component

Wireless networks particularly cellular networks are becoming a significant element for exchanging digitized data in low income countries. Nowadays, key sectors are influencing on cellular networks. For example: mobile financial transactions have gained success and healthcare centers are also providing healthcare services to patients remotely via mobile devices [50].

Wireless communication poses security concerns as compared to their wired counterpart. The same problem happens in wired networks such as the Internet. Internet is not secure medium of communication due to the openness and flexibility of its protocols [51]. Transmitting sensitive information from mobile devices to remote servers causes challenges of secure transmission. The most important security tool used for network communication is encryption [52, 53]. There are many different encryption algorithms each of them can be grouped into one of two categories as

Symmetric key cryptosystem or Public key cryptosystem. Encryption cannot protect information from eavesdropping and data modification during communication rather prevents information from extraction [52]. The General Packet Radio Service over Global System for Mobile applications (GPRS/GSM) wireless network secure the communication using mobile device authentication and user subscription to the mobile cellular operator using Subscriber Identity Module (SIM) [54].The most commonly used protocol to connect web-enabled mobile phones to remote servers is Wireless Application Protocol (WAP). Wireless Transport Layer Security (WTLS) is a layer within WAP that is responsible to secure the communication between the mobile phone and the server [55].

Mobile Information Device Profile (MIDP) is a specification published for the use of java on the family of cellular phones and simple pagers which have the least resources. It offers APIs for developing graphical user interfaces and establishing connectivity in J2ME. MIDP 2.0 is the latest version that was released in November 2002 [56] and the first MIDP 2.1 model in 2007. MIDP 2.0 implements secure connection by one or more of the following: HTTPS over TLS, SSL version 3, WTLS and WAP TLS profile and Tunneling [56]. Java enabled cellular phones in the world implemented with MIDP1.0 do not have secure wireless connection. Also cellular phones with MIDP 2.0 WTLS may not be applicable for some scenarios. For example: users cannot change the standard encryption key based on the sensitivity of their data. It is also not applicable in message oriented applications like SMS.

The handshake used in a WTLS connection takes a lot of time to complete the number of transactions. In addition to that a particular certificate is around 1K which takes a large portion in resource limited devices. Only the device manufacturers and their operating system vendors can control it. Symmetric key cryptosystem is known that it is used for simple communication and least intensive in terms of complexity as compared to public key cryptosystem. It is more suitable on mobile devices once the client and server share the secret key; it will be more efficient than using HTTPS.

The performance evaluation made on the six most common symmetric algorithms for power consumption, time consumption and throughput [57], those are: AES (Rijndael), DES, 3DES, RC2, Blowfish and RC6. Blowfish has better performance than others with low time

consumption, high throughput and low power consumption in both the encryption and decryption times.

Though public key cryptography (Asymmetric cryptography) is more complicated and much slower than the symmetric cryptography, it addresses the concern of key exchange of symmetric cryptography which allows secure communication over the insecure channel (Internet). The most commonly known asymmetric cryptography algorithms are: RSA (Ron Rivest, Adi Shamir, and Leonard Adleman), the most commonly used asymmetric algorithms: DH (Diffie-Hellman) key agreement protocol that is used for key exchange and the relatively newer technology ECC (Elliptic Curve Cryptography).

Like in RSA, ECC is used for secure key encryption, distribution and digital signatures. ECC is such an excellent choice for doing asymmetric cryptography in the devices having limited storage, processing capacity, bandwidth, and power supply like cellular telephones and the newer wireless devices where efficiency of resource is critical. ECC algorithm requires a smaller percentage of resources required by RSA and others. ECC use more complex algorithm than RSA but it requires much shorter keys for the same level of security achieved by RSA. ECC [58] with 160-bit key has about the same level of security as RSA with 1024-bit key. ECC keys in turn makes the cryptographic operations to run on the resource limited devices with fewer processor cycles, faster operation which in turn consumes less memory and reduces power consumption while still keeping equivalent security. ECC [59] is suitable for mobile devices since it is faster and power efficient than RSA.

This thesis work also came through the implementation of ECC and Blowfish algorithms to ensure that the private information flow on the pervasive healthcare system is secured. Once the secret key of the blowfish algorithm is shared using ECC public key encryption, every transaction of private information is encrypted using blowfish algorithm. Information of Patients which is sent from physician's terminal or patient's terminal will be encrypted using the secret key of the owner. At the destination terminal the encrypted message will be decrypted using the same shared secrete key.

The ECC encryption component physically resides at patient terminal on the secret key encryption and sender service component. The ECC decryption component physically resides at physician terminal on the Encrypted secret key listener and decryption service in order to get

secret key of that particular patient. The Blowfish encryption component physically resides at patient terminal on the Health status encryption service and at the physician terminal on the Message editor and encryption service. The Blowfish decryption component physically resides at patient terminal on the Message decryption service and at the physician terminal on the Message decryption service.

5.1.2.2 Users Access Rights and Policy Manager Component

The owners of medical information have a right of granting and inhibiting access rights to individuals on their medical information. Patients can define the access rights for those that can be in need of patients' medical records. The owner of the medical records defines access right policy on his/her medical resources for each user. For a particular user, owner defines how a user can get access on each kind of disease information, what attributes are accessible for that particular disease and expire date of the granted access right. Some of the attributes of a patient profile are those that uniquely identify the owner such as name, phone number and house number. Other attributes such as date of birth, sex, country, city, kebele has less probability of identifying the owner of the information uniquely. According to the need of the owner of the medical information the patient identifier attributes can be coded while users are getting access. Patient identifier attributes are coded if the patient is privacy concerned. A simple technique is used to encode the name and house number of the patients by randomly generated three English alphabets followed by three randomly generated integer numbers (Appendix G). Managing privacy protections at the attribute level is much flexible and satisfies the need of the owners of medical information for sharing their information with other users. As the result, roles of the users of medical information are not static since the owner can change the privacy protection setting any time and at the same time the roles are as many as the possible combinations of the different attributes. The policies made by the owners of medical information are used to express their security requirements as well as the respective security mechanisms selected by them are used to enforce the requirements.

Access right policy is made by patients on the attributes such as Name, date of birth, sex, phone number, House number, Kebele, Sub-City, City, Country, disease name, treatment taken, name of the physician by whom the patient has seen (Appendix G, I). Patient can set access right policy for each user of medical information on his/her resources. In this context resources are

medical information mainly disease name, treatment and other related information. For a given user of medical records, the owner of medical information can provide different combination of accessible attributes for each resource depending on the sensitivity of the resource in order to satisfy the required anonymity. However, for the users such as court of law who cannot be limited by the policy set by the owners will help the physicians to know the consent (policy) made by the owners and takes an appropriate action after the disclosure. If the policy set by a particular patient for court of law is against the request of the court, physicians must inform the owners of the disclosure. Otherwise it may not be important to inform about the disclosure. For each granted access right, date of grant and expire date of access are given by the owner of the resource from his/her mobile device and kept in the remote database. This component physically resides at patient mobile terminal at the Patient privacy protection policy sender service component.

5.1.2.3 Granularity Awareness Component

Access rights on medical information are granted by its owners. Medical information and the attributes that identify the owner are grained down in such a way that it is suitable to classify them as sharable and non-sharable according to their need. The grained down patient identifier attributes are: name of the patient, phone number, house number, kebele, sub city, city, country. Primarily the name and phone numbers uniquely identify patients, and house number can also uniquely identify patients with the help of additional attributes such as date of birth, sex, kebele, sub city, city, etc. These attributes should be classified as sharable or no-sharable for each medical resource by the owners. According to the grained down information patients can define policies in order to anonymize their identity before any sharing of information is made with others. This component physically resides on the patient terminal.

5.1.2.4 Constraints and Permissions Component

Access rights of the users are constrained by date of grant and expire date. The owners have a right to set expire date for each user and in turn for each kind of medical resource in order to inhibit access after the day of the expire date. A particular user of medical information may have different expire date for each kind of medical resource with different combination of attributes of a particular patient. The expire date is the policy made by each patient on their own medical information and also can be changed anytime according to the need of the owner whether to

extend or shorten the time of access for the users. This component physically resides on patient terminal.

5.1.2.5 Data Persistence Control Component

The system gets CD4 status messages each time from HIV patients who check their CD4 count periodically. Each time CD4 status message is sent to the health care unit and stored in the database if the CD4 count of the patients is less than or equal to 350 cells/mm^3. A huge amount of records are going to be on the database and at the same time patients' detailed medical information is going to be available. However, the data is very huge and not suitable to be processed by mobile devices. Thus, the data is controlled by an executable file which deletes the data periodically if the data is not important for the communication process between physicians and their respective patients. The executable file checks periodically for the CD4 status messages stored in the database and which of them are not needed farther by both patients and their respective physicians. Once such messages are found it automatically stores the important messages only as a medical history and deletes the huge data used for processing the communication. The time to trigger the executable file is set to be every 24 hours. The executable file triggers periodically as per the setting; 24 hours. This component physically resides in the remote server.

5.1.2.6 Classification of Resource Component

Medical information of the patients is grained down in the way that is suitable to assign access right at the attribute level. Once a user is selected for access grant in some medical resources, owner should classify the attributes as sharable/not sharable for each medical information to that particular user. A patient can grant some of the attributes of his/her information and inhibit others. Such classification of the attributes is important to achieve the required anonymity of the owners during medical information sharing. Resource classification by itself is a policy developed by the owners of medical information. This component physically resides on the patient terminal.

5.1.3 The Healthcare Module

This module is a healthcare web server that provides health related services. It provides service to the patients as well as their physicians. Patients and their physicians get access to the

webserver through their smartphones. This consists of the apache tomcat server to run the servlet codes and the database for storing information. In this subsection we will see the detail architecture of the healthcare module as shown in Figure 5-8. It consists of the web applications and the database component. In subsequent subsections, the functions of each component will be explained in detail.

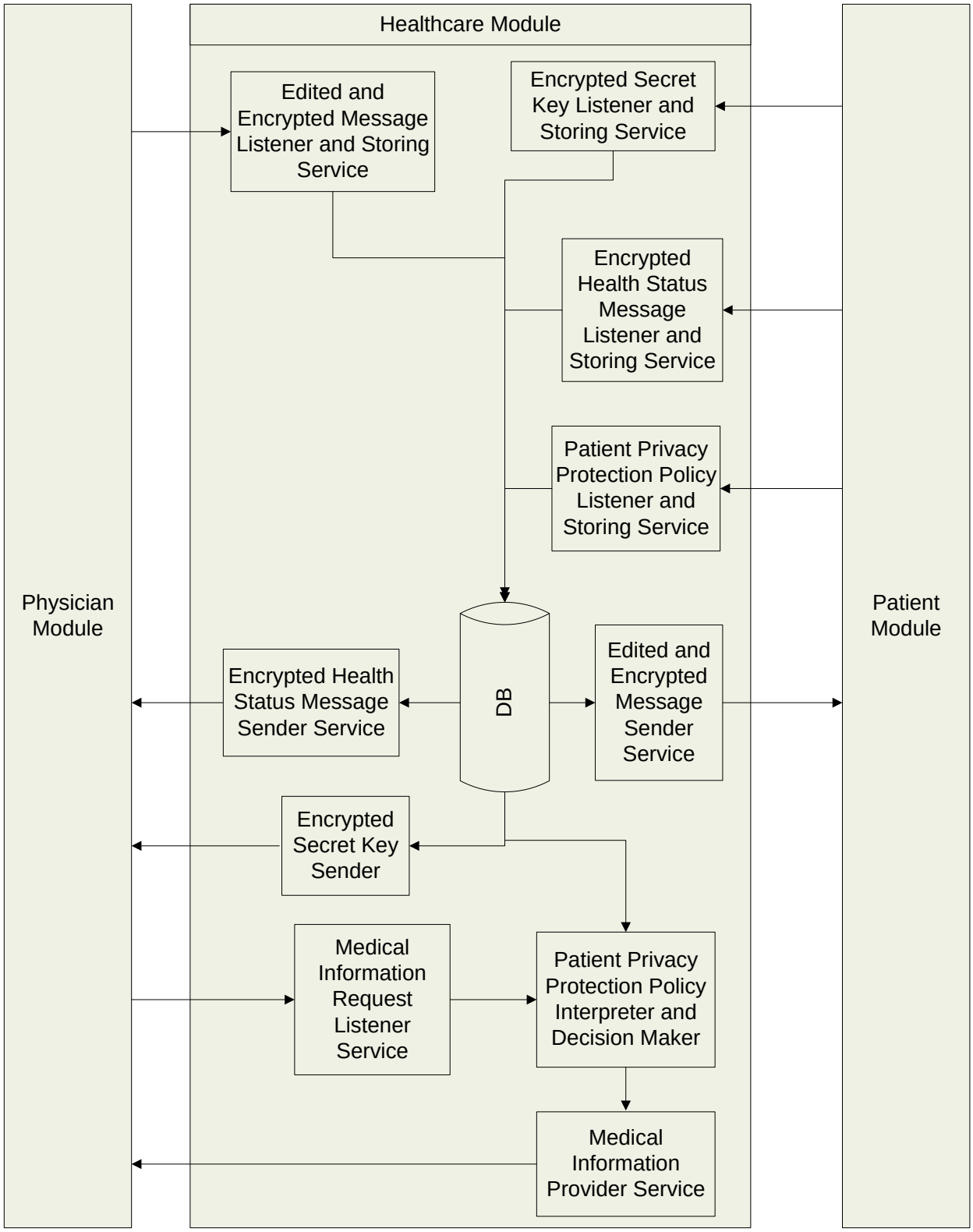


Figure 5-8: *Architecture of the healthcare module*

5.1.3.1 Encrypted Secret Key Listener and Storing Service Component

The secret key which is chosen by a patient for the blowfish encryption/decryption technique is sent to the respective physician via the healthcare unit. Once it is arrived in the server, it is stored in the database to be shared by that particular physician.

5.1.3.2 Encrypted Secret Key Sender Component

The secrete key stored in the database will be sent to the respective physician for decryption/encryption purpose while communicating with the owner of the secret key.

5.1.3.3 Encrypted Health Status Message Listener and Storing Service Component

The health status messages coming from patients are encrypted by their own secret key and sent to the respective physicians. This component receives an encrypted message and store in the database until the respective physician edits and sends it back.

5.1.3.4 Encrypted Health Status Message Sender Service Component

This component provides encrypted health status messages of patients to the corresponding physicians. When request comes from the physicians' terminal, it is responsible to deliver the messages accordingly.

5.1.3.5 Edited and Encrypted Message Listener and Storing Service Component

Once the status messages are decrypted on the physician terminal, the messages are edited accordingly, encrypted and sent to the patients back. The messages are delivered via the web service and received at this component and stored in the database to be available for the owners.

5.1.3.6 Edited and Encrypted Message Sender Service Component

The edited messages are fetched from the database and sent to the respective patient's terminal by this component. When a patient initiates his/her mobile terminal, this component starts sending the messages that has been sent to him/her.

5.1.3.7 Patient Privacy Protection Policy Listener and Storing Service Component

Patients have the right to set policies on their medical resources. The policies will be sent to the healthcare module and stored in the database and are editable at any time by the owner of the

policies. The policies are received at this component of the healthcare module and stored in the database.

5.1.3.8 Medical Information Request Listener Service Component

Physicians may send access request on medical information for different purposes. The request can be for physician use or for other users use. The request is sent to the healthcare module and received at this component. It passes the request to the policy interpreter and decision maker component before providing any medical information.

5.1.3.9 Patient Privacy Protection Policy Interpreter and Decision Maker Component

If the request if for information sharing purpose, this component identifies who the requester is and which patient(s) resource is required. It gets the policies set by the respective patient for that particular user from the database in order to interpret. Accordingly decision is made on what resources are allowed for the user to access and in what way. Finally it passes the allowed medical resources to the next component. If the request is for the use of the respective physician, it provides the required information accordingly.

5.1.3.10 Medical Information Provider Service Component

The allowed medical resources of patients are received from the policy interpreter and decision maker component and sent to the respective physician terminal via this component.

5.1.3.11 Database Component

This component is a MySQL database server which stores the data forwarded by the different services and provides data for the data provider components in the healthcare module.

5.1.4 The Physician Module

This module includes the physician's smartphone that communicates with the healthcare system to get health status of patients and send back message to the patients accordingly. The physician module also includes the decision maker component that helps physicians how to share patients' medical information with other stakeholders. In this subsection the components of the physician module are explained in detail. The detail architecture of this module and its interaction with the healthcare module is as shown in Figure 5-9.

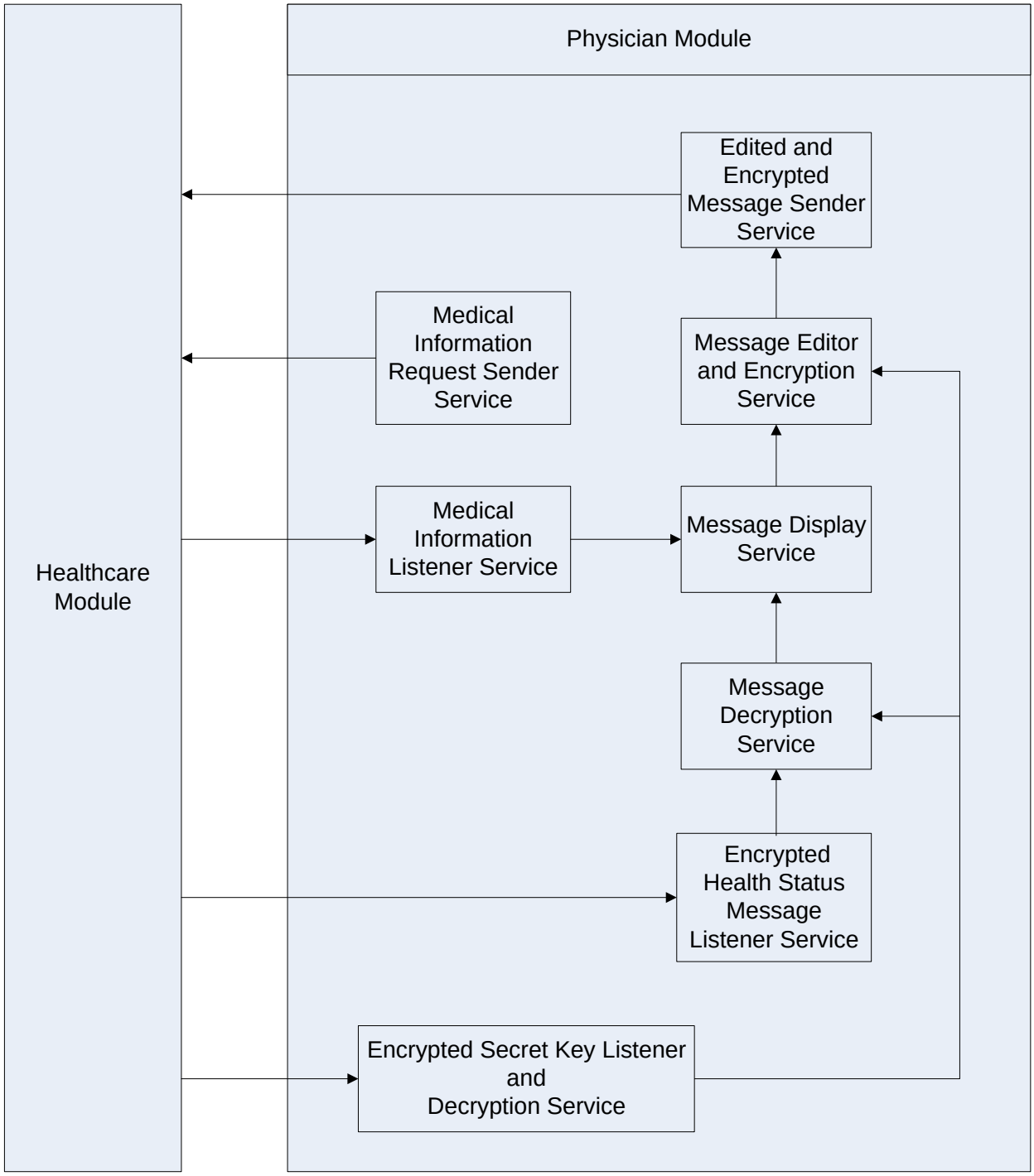


Figure 5-9: *Architecture of the physician module*

5.1.4.1 Encrypted Secret Key Listener and Decryption service Component

Physicians get secret key of each patient that are encrypted by their public key of the ECC then received at this component in order to be decrypted by their corresponding private key. The

secret keys that are shared among physicians and patients are using an Elliptic Curve algorithm (ECC). Each key is an input to the Blowfish encryption and decryption techniques. A secret key of a patient is required at the physician terminal in order to decrypt the encrypted messages coming from that particular patient. It is also important to encrypt the edited messages and send it back to that particular patient.

5.1.4.2 Encrypted Health Status Message Listener Service Component

The encrypted health status messages coming from patients are received at this component. It passes the coming messages to the next component for decryption.

5.1.4.3 Message Decryption Service Component

The encrypted messages coming from patients are decrypted by blowfish symmetric cryptosystem using the shared secret key. Then the original message will be available for the physician.

5.1.4.4 Message Display Service Component

It is a graphical user interface used to display decrypted messages that come from patients to a physician. It is also used to display medical information retrieved from the healthcare unit according to the request of the physician.

5.1.4.5 Message Editor and Encryption Service Component

Based on the health status messages come from patients, a physician can make changes on the messages in the way that is suitable to inform patients and give appointments for them. Then the edited messages are going to be encrypted using blowfish symmetric cryptosystem before leaving the physician terminal.

5.1.4.6 Edited and Encrypted Message Sender Service Component

Once the edited message is encrypted, it is going to be sent to the healthcare unit via web service. When it reaches at the healthcare unit it will be stored in the database to be available for the respective patient.

5.1.4.7 Medical Information Request Sender Service Component

When a physician needs to know medical history of a particular patient of him/her, to do some analysis on the detail medical information of the patients or when physician received a request

on medical information from other users, this component sends physicians request to the healthcare unit by specifying identity of the requester.

5.1.4.8 Medical Information Listener Service Component

Once the medical information request is sent to the healthcare unit, it is received by the medical information request listener component and processed by the patients' privacy protection policy interpreter and decision maker component. If the request is for physician use such as getting medical history of a particular patient or detail medical information of the respective patients, it provides the required medical history of the patient via medical information provider component or sends some notification about the successful export of the detail medical information of the patients to his active directory. If the request is for sharing medical information with other users, accordingly the medical information will be sent to the physician terminal and received at this component.

5.1.5 The Other Users of Medical Information

Medical information of patients can be used by healthcare providers for treatment purposes. In addition to that medical records can also be required by other stakeholders for different purposes and physicians are responsible to provide medical information to them according to the privacy protection policy of the patients. But there are cases that do not depend on the patients privacy protection policy as already mentioned on the medical ethics, in such cases physicians provide medical information in the way that is required and inform patients of the disclosure and reasons for it. Types of such users are listed below:

- Family
- Medical Researchers
- Court of Law
- Payer Organizations (Insurance,..)
- Public Health Management
- Public Policy Makers
- Hospital (Hospital Accreditations)

5.2 Summary

In this chapter, we have seen the details of the overall system components. The system consists of five main modules: the patient module, the patient privacy protection manager module, the healthcare module, the physician module and the other users module. Each component of the system is discussed in detail. The patient module includes the Moxi z to find the CD4 count of the patient then send to the patient's smartphone via Bluetooth in order to process the reading. At the patient mobile terminal the information is processed then generates a message that shows status of the patient. Finally the message is encrypted and sent to the respective physician via the http communication protocol. At the same time the patient can set privacy protection policy on his/her resources from remote. The other component is the privacy protection manager module that consists of the sub-components: the secret key sharing, the message encryption and decryption components and other privacy protection techniques. Each of the sub-components physically appear in the patients' and/or the physicians' mobile terminals. The third module is the healthcare unit which consists of: the servlet programs for the web based application, the database component to store medical information of patients, shared secret key and patients' privacy protection policies. The fourth module is the physician module which enables physicians to monitor their patients' health status remotely by decrypting the messages sent from the patients and encrypts messages before sent to the respective patients. There are secondary users of medical information who get information based on the consents of the owners. The users get benefit from the Privacy Aware Pervasive Healthcare System via physicians.

Chapter 6 : The Prototype Implementation

In Chapter five, we have already seen the components of the PAPHS in detail. In this chapter we will see how the system works practically using prototype implementation. First we will present the tools and technologies used to implement and demonstrate the system. Then we will present how the components of the system are implemented. We have also prepared different scenarios in order to show the readers how the PAPHS works. Finally, we will demonstrate our prototype implementation using screen shots.

6.1 The Tools and Technologies used for the Implementation

For the prototype implementation of the system the following tools and technologies are used:

1. *MySQL database server version 5.0.2:* to store detail medical information of patients and privacy protection related information of patients.
2. *Apache Tomcat version 6.0.37:* is used for HTTP web server environment in order to run the java servlet codes.
3. *CLDC Java(TM) platform micro edition SDK 3.0 (Java ME SDK 3.0):* is used as emulator platform and the DefaultFXPhone1[1] is used as a device (as smartphone) with the CLDC-1.1 device configuration and MIDP-2.0 device profile to implement the components of the patients and physicians.
4. *MySQL Connector/JDBC version 5.1.25:* is a native java driver which converts JDBC (Java Database Connectivity) calls into the network protocol used by the MySQL database.
5. *Moxi z automated cell counter*[2] *:* is used to count the CD4 of HIV patients.
6. *Bluetooth wireless communication technology*[3]*:* is used to communicate Moxi z with the smartphone of an HIV patient.
7. *Internet communication technology:* http protocol is used to communicate the smartphones with the server at remote area.

[1] due to budget limitation smartphone is implemented by a J2ME mobile phone emulator
[2] due to budget limitation it is implemented by a java class that generates a number within the range of a valid CD4 count
[3] the Bluetooth communication between the Moxi z and the smartphone is emulated by the two synchronized functions the one generates a valid CD4 count and the other gets the CD4 reading

6.2 Implementation

The prototype implementation of the system mainly consists of five modules. The implementation architecture of the PAPHS is as shown in Figure 6-1. Some of the sub components of the privacy protection manager module are implemented at the patients' module and the physicians' module. The Other Users module belongs to the secondary users of medical information. The users have direct contact with physicians (responsible body) in order to get medical information from the PAPHS according to their request and the consent of the owners of the information. The detail implementation of each component will be explained in subsequent subsections using algorithms.

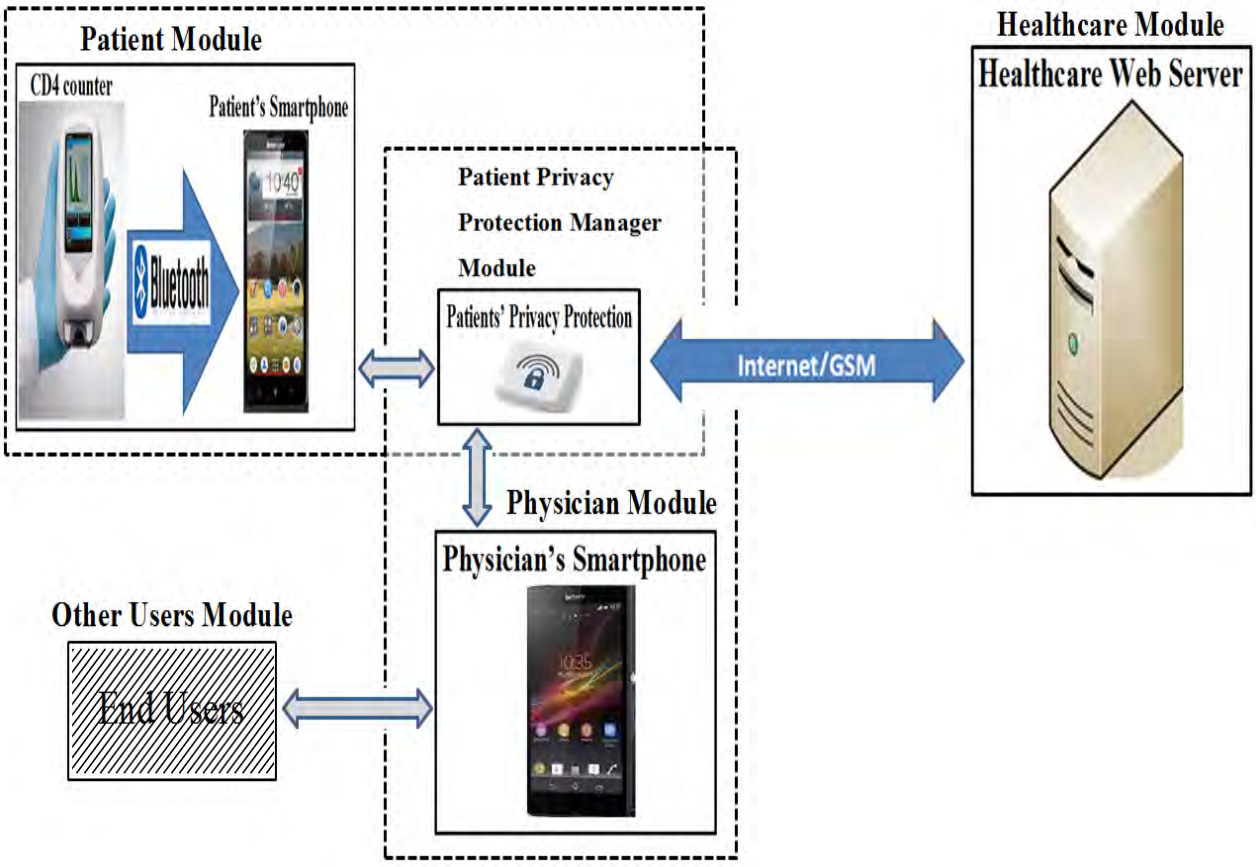


Figure 6-1: *Implementation diagram of the PAPHS*

6.2.1 Components of the Patient Module

- ***Moxi z Automated Cell Counter Component***: it is emulated by a J2ME thread that consists of two synchronized functions. The first function is responsible to generate a valid CD4 count and provides to the second function. The thread is used in order to run the Moxi z emulator periodically. Algorithm 6.1 explains how the Moxi z emulator is done as shown in Figure 6-2 .

Algorithm 6.1

Input - three randomly generated CD4 counts < = 1600

Output - provide the three randomly generated CD4 counts to the listener component(the other function)

1. Maximum size of the vector is 3
2. Declare the vector as Vector **CD4Message** = new Vector()
3. *Synchronized* **putMessage** ()
   a. While CD4Message is 3
      I. Wait() function is called until the control is released by the other *Synchronized* **getMessage**() function
   b. Generate a random integer number with in the range of valid CD4 counts
   c. Assign the number to the variable as int **CD4** = new Random(). nexInt(1600)
   d. Add the CD4 to the vector **CD4Message** as: **CD4Message.**addElement(**CD4**)
   e. If the vector is full
      I. Call notify() in order to give the control to the CD4 count listener component **getMessage**()

Figure 6-2: *Moxi z Automated Cell Counter Algorithm*

- ***CD4 Count Listener Component***: it is the second synchronized function which gets CD4 reading from the first function and provide to the other thread in which the reading is interpreted. Algorithm 6.2 contains a set of instructions that emulate CD4 count listener as shown in Figure 6-3.

Algorithm 6.2

Input - gets three CD4 count from **putMessage**() synchronized function

Output - provide the three CD4 count to the Data Interpreter component

1. *Synchronized* **getMessage**()is a CD4 count listener
   I. Notify() the *synchronized* function **putMessage**() that the control is here
   II. While the vector is empty
      a. Call Wait() to notify this function is releasing the control since no CD4 count to read
   III. If the vector has >= 1
      a. Get the first element from the vector
      b. Convert the element to an Integer then assign to a integer variable **CD4count**
      c. Remove the element from vector
      d. Return the value on **CD4count** variable
2. The **CD4count** will be passed to the next component (Data Interpreter)

Figure 6-3: *CD4 Count Listener Algorithm*

- ***Data Interpreter and Health Status Message Generator Service Component***: this component is also implemented by the J2ME thread that gets the CD4 count from the above component. The CD4 reading is preprocessed in this component in order to generate the appropriate message that is going to be sent to the respective physician. Algorithm 6.3 contains a set of instructions that implements the data interpreter and health status message generator component as shown in Figure 6-4.

Algorithm 6.3

Input - gets a CD4 count from **getMessage**() synchronized function
- **nameofUser** contains name of the patient

Output - return the equivalent health status message for the first CD4 count that is <=350

1. IF the CD4 is less than or equals to 350
   a. IF the CD4 is < 200
      i. If the CD4 is < 75
         Message = The CD4 count of "+**nameofUser**+" has reached "+ CD4 count +" cells/mm3, check for AIDS and Mycobacterium avium complex[MAC]
      ii. Else If the CD4 is < 100
         Message = The CD4 count of "+**nameofUser**+" has reached "+ CD4 count +" cells/mm3, check for AIDS and for Toxoplasmosis and Cryptococcosis
      iii. Else
         Message = The CD4 count of "+**nameofUser**+" has reached "+ CD4 count +" cells/mm3, check for AIDS and for Pneumocystis pneumonia[PCP]
   b. ELSE IF the CD4 is < 300
      Message = The CD4 count of "+**nameofUser**+" has reached "+ CD4 count +" cells/mm3, Please initiate ART"+" and also initiate ARV to prevent PCP
   c. ELSE
      Message = The CD4 count of "+**nameofUser**+" has reached "+ CD4 count +" cells/mm3, Please initiate ART
2. ELSE
   Sleep this thread for a while until the next randomly generated valid CD4 count is fetched from the vector via the **getMessage** *synchronized* function

Figure 6-4: *Data Interpreter and Health Status Message Generator Algorithm*

- ***Health Status Encryption Service Component***: this is implemented in the same J2ME thread in order to encrypt the message generated in the above component using blowfish (Appendix B). Algorithm 6.4 contains a set of instructions that implements the health status message encryption component as shown in Figure 6-5.

Algorithm 6.4

Input - the message to be encrypted **message**
- the secret key **K** in bytes
- import the package **org.bouncycastle.crypto**

Output - A byte [] variable **result** to keep the encrypted message
(cipher text)

1. Declar an object for the BufferedBlockCipher and the KeyParameter as:
   - BufferedBlockCipher **cipher**
   - KeyParameter **key**
2. Create the object **cipher**
3. Create the object **key** by providing the secret key **K** as an argument;
   key = new KeyParameter (**K**)
4. Pass the secret key **K** as an argument to the function init of the instance **cipher** as **cipher**.init(true, **K**)
5. Get the size of the output cipher text of the **message (**in bytes)
6. Declare the byte array variable called **result** having the size of the output cipher text
7. Encrypt the **message** then the returned value will be assigned to the variable **result**

Figure 6-5: *Health Status Encryption Algorithm*

- ***Encrypted Health Status Message Sender Service Component***: this is implemented using http communication protocol in the same J2ME thread in order to send the encrypted message to the remote server. Before the encrypted message is sent to the server, it is converted into base 64 encoding. Algorithm 6.5 contains a set of instructions that implements the encrypted health status message sender component as shown in Figure 6-6.

Algorithm 6.5

Input - the encrypted message **encryptedSMS** (bytes array)
- the phone number of the patient **patientphoneNo**
- the phone number of the respective physician **physicianphoneNo**
- import **org.bouncycastle.util.encoders.Base64** for encoding

Output - Message **delivery notification**

1. Create an instance **c** of an HttpConnection to the remote server using the url address of the web application that listens the parameters will be sent ;
   HttpConnection c = Connector.open (http://localhost:8080/sendMessage/getConnection)
2. Set the request property of the instance **c**
3. Set the request method of the instance **c** to be a POST httpConnection
4. Create an instance of a DataOutputStream for the httpConnection instance **c** as;
   DataOutputStream **os** = **c** . openDataOutputStream()
5. Convert the **encryptedSMS** (in byte array) to its base64 equivalent before sent to the remote server to be stored;
   byte[] **EncB64SMS = Base64.encode (encryptedSMS)**
6. Then convert the **base64** equivalent to **string** before sending the message; String **encryptedData** = new String (**EncB64SMS**)
7. Then send the message, phone number of the patient and the respective physician
   a. **os**.writeUTF (**encryptedData**)
   b. **os**.writeUTF (**patientphoneNo**)
   c. **os**.writeUTF (**physicianphoneNo**)
   d. Close the object **os**
8. To get the message delivery notification
   a. create an instance of a DataInputStream **is** using the http connection instance **c** ; DataInputStream **is** = **c**.openDataInputStream()
   b. create a stringBuffer **sb** to accumulate the stream data of the notification; stringBuffer **sb =** new stringBuffer()
   c. declare an integer **ch** that reads each character in the stream; **ch = is**.read()
9. WHILE **ch** is different from -1 (end of the data stream)
   a. Change **ch** (the decimal equivalent of a character) to the equivalent character
   b. Then append the character to the **sb** string buffer ; **sb . append**(char(ch))
10. END OF WHILE
11. Display the notification sent from the server(**sb**)

Figure 6-6: *Encrypted Health Status Message Sender Algorithm*

- ***Incoming Edited and Encrypted Message Listener Service Component***: the edited massage is fetched by the J2ME thread via the http connection protocol from the remote database and stored in a string buffer before entered into the decryption service. Algorithm 6.6 contains a set of instructions that implements the encrypted message listener component as shown in Figure 6-7.

Algorithm 6.6

Input - the phone number of the patient **patientphoneNo**
- the phone number of the respective physician **physicianphoneNo**

Output - **EncryptedMessage (**in byte array**)** from physician

1. Create an instance **c** of an HttpConnection to the remote server using the url address of the web application that listens the parameters will be sent ;
   HttpConnection c = Connector.open (http://localhost:8080/sendMessage/pmessage)
2. Set the request property of the instance **c**
3. Set the request method of the instance **c** to be a POST httpConnection
4. To get the **EncryptedMessage** from the remote server
   a. Create an instance of a DataOutputStream for the httpConnection instance **c** as;
      DataOutputStream **os** = **c** . openDataOutputStream()
   b. Then send the phone number of the patient and the respective physician
      I **os**.writeUTF (**patientphoneNo**)
      II **os**.writeUTF (**physicianphoneNo**)
      III Close the object **os**
   c. Create an instance of a DataInputStream **is** using the http connection instance **c** ;
      DataInputStream **is** = **c**.openDataInputStream()
   d. create a stringBuffer **sb** to accumulate the stream data of the incoming message;
      stringBuffer **sb =** new stringBuffer()
   e. declare an integer **ch** that reads each character in the stream; **ch = is**.read()
5. WHILE **ch** is different from -1 (end of the data stream)
   a. Change **ch** (the decimal equivalent of a character) to the equivalent character
   b. Then append the character to the **sb** string buffer ;
      sb.append(char(ch))
6. End of WHILE

7. Declare a String variable **SMS** to hold the message in the string buffer as: **SMS** = **sb**.toString()
8. The **SMS** data is a string data that was converted from base64 encoding which is made for storing purpose
9. So, convert the **SMS** back to base64;
    a. byte[] **base64SMS = Base64.decode (SMS.getBytes (UTF-8))**
    b. assign to the **EncryptedMessage; EncryptedMessage = base64SMS**

Figure 6-7: *Incoming Edited and Encrypted Message Listener Algorithm*

- ***Message Decryption Service Component***: this is implemented using J2ME thread that gets the encrypted messages from the remote server via http communication protocol. Once the message is arrived it is going to enter into the blowfish decryption code in order to find the original message (Appendix B). Algorithm 6.7 contains a set of instructions that implements the message decryption component as shown in Figure 6-8.

Algorithm 6.7

Input - Encrypted message **cipherMessage**
- The secret key **K** in bytes
- import the package **org.bouncycastle.crypto**

Output - A String variable **result** to keep the decrypted message (Original message)

1. Declar an object for the BufferedBlockCipher and the KeyParameter as:
    - BufferedBlockCipher **cipher**
    - KeyParameter **key**
2. Create the object cipher
3. Create the object key by providing the secret key **K** as an argument; **key** = new KeyParameter (**K**)
4. Pass the secret key **K** as an argument to the function init of the instance **cipher** as **cipher**.init(false, **K**)
5. Get the size of the output text of the **cipherMessage** (in bytes)
6. Declare the byte array variable called **result** having the size of the output text
7. Decrypt the **cipherMessage** then the returned original message is converted to String and assigned to the variable **result**

Figure 6-8: *Message Decryption Algorithm*

- ***Message Display Service Component***: the J2ME MIDlet classes are used for the development of graphical user interface (GUI) on which to display the messages. Algorithm 6.8 contains a set of instructions that implements the message display component as shown in Figure 6-9.

Algorithm 6.8

Input - Decrypted **messages** (original **SMS**)
- import the package **javax.microedition.midlet**
- import the package **javax.microedition.lcdui**

Output - the **SMS** on the mobile of the patient

1. Create the Display object on which the form lies
2. Initialize display
3. Create the form and give title for the form
4. Create the command and add the command on the form
5. Write a set of codes for the command action
   a. keep decrypted messages (**SMS**) arrived in the patient mobile in a String array **messages**
   b. for the number of messages **N**
   c. for i=0, i < **N**
   d. Display each message as preceded by number; i+1. **Messages** [i]

Figure 6-9: *Message Display Algorithm*

- ***Patient Privacy Protection Policy Sender Service Component***: it is implemented as a J2ME thread which uses http protocol in order to send the privacy policy of a particular patient to the remote server to be stored in the database. Algorithm 6.9 contains a set of instructions that implements the patient privacy protection policy sender component as shown in Figure 6-10.

Algorithm 6.9

Input - Task identification **id**
- Name of the end user **Euser**
- Phone number of the patient **patientphoneNo**
- Phone number of the respective physician **physicianphoneNo**
- name of disease (information) **disease**
- patient privacy protection code **encryptedidentity**
- Patient privacy protection **preferenceIndicator**
- **age** attribute
- **sex** attribute
- **country** attribute
- **city** attribute
- **kebele** attribute
- data of expire for access right **expireDate**

Output - **Policy** delivery notification

1. Create an instance **c** of an HttpConnection to the remote server using the url address of the web application that listens the parameters will be sent ;
   HttpConnection c = Connector.open (http://localhost:8080/sendMessage/pmessageNo)
2. Set the request property of the instance **c**
3. Set the request method of the instance **c** to be a POST httpConnection
4. To send the **policy** and get the **Policy** delivery notification from the remote server
   a. Create an instance of a DataOutputStream for the httpConnection instance **c** as;
      DataOutputStream **os** = **c** . openDataOutputStream()
   b. According the policy set by the patient using the graphical user interface
   c. If the patient is privacy concerned on the particular disease, code will be generated
   d. The code generated is based on the randomly generated three English alphabets followed by the three digit integers
   e. Then send the phone number of the patient and other parameters of the policy
      - **os**.writeUTF(**id**)
      - **os**.writeUTF(**Euser**)
      - **os**.writeUTF (**patientphoneNo**)
      - **os**.writeUTF (**physicianphoneNo**)
      - **os**.writeUTF(**disease**)

```
        - os.writeUTF(encryptedidentity)
        - os.writeUTF(preferenceIndicator)
        - os.writeUTF(age)
        - os.writeUTF(sex)
        - os.writeUTF(country)
        - os.writeUTF(city)
        - os.writeUTF(kebele)
        - os.writeUTF(expireDate)
        - Close the object os

   f. Create an instance of a DataInputStream is using the http
      connection instance c ;
      DataInputStream is = c.openDataInputStream()
   g. create a stringBuffer sb to accumulate the stream data of
      the incoming notification message; stringBuffer sb = new
      stringBuffer()
   h. declare an integer ch that reads each character in the
      stream; ch = is.read()
5. WHILE ch is different from -1 (end of the data stream)
   a. Change ch (the decimal equivalent of a character) to the
      equivalent character
   b. Then append the character to the sb string buffer ;
      sb . append(char(ch))
6. End of WHILE
7. Display the notification by converting the buffer sb to
   string; sb.toString()
```

Figure 6-10: *Patient Privacy Protection Policy Sender Algorithm*

- ***Secret Key Encryption and Sender Service Component***: this component is done based on the recent work of Belaynew [60]. A J2ME thread is used to encrypt a 64 bit secret key by ECC using a public key of the respective physician and sent to the remote server in order to be available for the respective physicians. The 64 bit secret key is also going to be used for blowfish encryption and decryption code that are implemented at the patients' mobile terminals. Algorithm 6.10 contains a set of instructions that implements the secret key encryption and sender component as shown in Figure 6-11.

Algorithm 6.10

Input - The secret key **K** in bytes
- Public key of the respective physician **Pphy**
- Phone number of the patient **patientphoneNo**
- Phone number of the respective physician **physicianphoneNo**
- import the package **org.bouncycastle.crypto**

Output - Secret key delivery notification

1. Encrypt the secret key using **ECC** public key crypto system
2. Declare a byte array **enckey** to save the encrypted key using the public key of the respective physician as; **enckey** = **ECC** (**k, Pphy**)
3. Encode the result in Base64 then convert to its String equivalent as ; String **encryptedkey = new String (Base64(enckey))**
4. Create an instance **c** of an HttpConnection to the remote server using the url address of the web application that listens the parameters will be sent ; HttpConnection c = Connector.open (http://localhost:8080/sendMessage/secretkeylistener)
5. Set the request property of the instance **c**
6. Set the request method of the instance **c** to be a POST httpConnection
7. To send the encrypted secret key and get the delivery notification from the remote server
   a. **os**.writeUTF(**encryptedkey**)
   b. **os**.writeUTF(**patientphoneNo**)
   c. **os**.writeUTF(**physicianphoneNo**)
   d. Close the object **os**
   e. Create an instance of a DataInputStream **is** using the http connection instance **c** ; DataInputStream **is** = **c**.openDataInputStream()
   f. create a stringBuffer **sb** to accumulate the stream data of the incoming notification message; stringBuffer **sb =** new stringBuffer()
   g. declare an integer **ch** that reads each character in the stream; **ch = is**.read()
8. WHILE **ch** is different from -1 (end of the data stream)
   a. Change **ch** (the decimal equivalent of a character) to the equivalent character
   b. Then append the character to the **sb** string buffer ; **sb . append**(char(ch))
9. End of WHILE
10. Display the notification in **sb** in string format

Figure 6-11: *Secret Key Encryption and Sender Algorithm*

6.2.2 Components of the Healthcare Module

- ***Encrypted Secret Key Listener and Storing Service Component***: it is implemented by a java servlet code that gets the secret key and the phone number of the owner of the key and the phone number of the physician with whom the patient needs to share. Then the encrypted key is stored in the database. Algorithm 6.11 contains a set of instructions that implements the encrypted secret key listener and storing component as shown in Figure 6-12.

Algorithm 6.11

Input - the encrypted key **encryptedkey**
- Phone number of the patient **patientphoneNo**
- Phone number of the respective physician **physicianphoneNo**

Output - Secret key storing notification

1. Create the DataInputStream **in** for the http servlet request sent from the patient mobile terminal
2. From the data input stream object **in** get the encrypted secret key then assign to **encryptedkey**
3. From the data input stream object **in** get the patient phone number then assign to **patientphoneNo**
4. From the data input stream object **in** get the respective physician phone number then assign to **physicianphoneNo**
5. Declare the database as **health** , user as **root** and the password of the database as **sa**
6. Create a connection object **conn** to the database from the java servlet using the **root, sa** and **health**
7. Create a statement object **stm** using the connection object **conn**
8. Write the query to insert the encrypted secret key **encryptedkey**, patient phone number **patientphoneNo** and physician phone number **physicianphoneNo**
9. Then execute the insert query statement using the object **stm** and get the **affected number of record**
10. IF the affected number of record is different from zero the secret key is stored. Then generate a notification and send it back to the patient using the http servlet response.

Figure 6-12: *Encrypted Secret Key Listener and Storing Algorithm*

- ***Encrypted Secret Key Sender Component***: it is also a java servlet code that retrieves the appropriate key and sends to the physicians' terminal when required for message encryption/decryption purpose. Algorithm 6.12 contains a set of instructions that implements the encrypted secret key sender component as shown in Figure 6-13.

Algorithm 6.12

Input - Phone number of the patient **patientphoneNo**
- Phone number of the respective physician **physicianphoneNo**

Output - the encrypted key **encryptedkey**

1. Create the DataInputStream **in** for the http servlet request sent from the physician mobile terminal
2. From the data input stream object **in** get the patient phone number then assign to **patientphoneNo**
3. From the data input stream object **in** get the respective physician phone number then assign to **physicianphoneNo**
4. Declare the database as **health** , user as **root** and the password of the database as **sa**
5. Create a connection object **conn** to the database from the java servlet using the **root, sa** and **health**
6. Create a statement object **stm** using the connection object **conn**
7. Write the query to fetch the encrypted secret key shared among a patient having a phone number **patientphoneNo** and the physician having a phone number **physicianphoneNo**
8. Declare a result set **ResultSet** to hold the returned row from the table
9. Execute the select query statement using the object **stm** and assign the row returned to the **ResultSet**
10. Get the secret key from the **ResultSet** then assign to the variable **encryptedkey**
11. Send the **encryptedkey** to the respective physician using the http servlet response

Figure 6-13: *Encrypted Secret Key Sender Algorithm*

- ***Encrypted Health Status Message Listener and Storing Service Component***: this is implemented by a servlet java program that uses http connection protocol in order to get secured messages from patients' mobile terminal and store the messages in the database. Algorithm 6.13 contains a set of instructions that implements the

encrypted health status message listener and storing component as shown in Figure 6-14.

Algorithm 6.13

Input - the encrypted message **encryptedData**
- the phone number of the patient **patientphoneNo**
- the phone number of the respective physician **physicianphoneNo**

Output - Message storing notification

1. Create the DataInputStream **in** for the http servlet request sent from the patient mobile terminal
2. From the data input stream object **in** get the encrypted message then assign to the **encryptedData**
3. From the data input stream object **in** get the patient phone number then assign to **patientphoneNo**
4. From the data input stream object **in** get the respective physician phone number then assign to **physicianphoneNo**
5. Declare the database as **health** , user as **root** and the password of the database as **sa**
6. Create a connection object **conn** to the database from the java servlet using the **root, sa** and **health**
7. Create a statement object **stm** using the connection object **conn**
8. Write the query to insert the encrypted message **encryptedData,** patient phone number **patientphoneNo** and physician phone number **physicianphoneNo**
9. Then execute the insert query statement using the object **stm** and get the **affected number of record**
10. IF the affected number of record is different from zero the encrypted message is stored. Then generate a notification and send it back to the patient using the http servlet response.

Figure 6-14: *Encrypted Health Status Message Listener and Storing Algorithm*

- ***Encrypted Health Status Message Sender Service Component***: this is implemented by a servlet java program which uses http protocol in order to send new messages came to the respective physicians' mobile terminal. The messages (SMS) are fetched from the database and sent without decrypting (Appendix F). Algorithm 6.14 contains a set of instructions that implements the encrypted health status message sender component as shown in Figure 6-15.

Algorithm 6.14

Input - the phone number of the physician **physicianphoneNo**

Output - Encrypted message **encrytedMessage**

1. Create the DataInputStream **in** for the http servlet request sent from the physician mobile terminal
2. From the data input stream object **in** get the physician phone number then assign to **physicianphoneNo**
3. Declare the database as **health** , user as **root** and the password of the database as **sa**
4. Create a connection object **conn** to the database from the java servlet using the **root, sa** and **health**
5. Create a statement object **stm** using the connection object **conn**
6. Write the select query to fetch the encrypted messages sent from the patients of the physician having a phone number **physicianphoneNo**
7. Declare a result set **ResultSet** to hold the returned messages from the table
8. Execute the select query statement using the object **stm** and assign the rows returned to the **ResultSet**
9. Get the encrypted messages of the patients from the **ResultSet** then assign to the variable **encrytedMessage**
10. Send the encrypted messages to the respective physician using the http servlet response

Figure 6-15: *Encrypted Health Status Message Sender Algorithm*

- ***Edited and Encrypted Message Listener and Storing Service Component***: this is implemented by a java servlet program that uses an http protocol in order to get messages edited by physicians and store the messages in the database. Algorithm 6.15 contains a set of instructions that implements the encrypted message listener and storing component as shown in Figure 6-16.

Algorithm 6.15

Input - the encrypted message **EditedencryptedData**
- the phone number of the patient **patientphoneNo**
- the phone number of the respective physician **physicianphoneNo**

Output - Message storing notification

1. Create the DataInputStream **in** for the http servlet request sent from the physician mobile terminal
2. From the data input stream object **in** get the encrypted message then assign to the **EditedencryptedData**
3. From the data input stream object **in** get the patient phone number then assign to **patientphoneNo**
4. From the data input stream object **in** get the respective physician phone number then assign to **physicianphoneNo**
5. Declare the database as **health** , user as **root** and the password of the database as **sa**
6. Create a connection object **conn** to the database from the java servlet using the **root, sa** and **health**
7. Create a statement object **stm** using the connection object **conn**
8. Write the query to update the existing encrypted message in the database by the edited and encrypted message **EditedencryptedData** that will be sent to the patient phone number **patientphoneNo** from the physician phone number **physicianphoneNo**
9. Then execute the update query statement using the object **stm** and get the **affected number of record**
10. IF the affected number of record is different from zero the edited and encrypted message is stored. Then generate a notification and send it back to the physician using the http servlet response

Figure 6-16: *Edited and Encrypted Message Listener and Storing Algorithm*

- ***Edited and Encrypted Message Sender Service Component***: this is implemented by a java servlet program which uses an http protocol in order to send the messages edited by physicians to the respective patients' mobile terminals. Algorithm 6.16 contains a set of instructions that implements the encrypted message sender component as shown in Figure 6-17.

Algorithm 6.16

Input - the phone number of the patient **patientphoneNo**

Output - Encrypted message **editedencrytedMessage**

1. Create the DataInputStream **in** for the http servlet request sent from the patient mobile terminal
2. From the data input stream object **in** get the patient phone number then assign to **patientphoneNo**
3. Declare the database as **health** , user as **root** and the password of the database as **sa**
4. Create a connection object **conn** to the database from the java servlet using the **root, sa** and **health**
5. Create a statement object **stm** using the connection object **conn**
6. Write the select query to fetch the edited and encrypted messages sent from the physician of the patient having a phone number **patientphoneNo**
7. Declare a result set **ResultSet** to hold the returned message from the database
8. Execute the select query statement using the object **stm** and assign the row returned to the **ResultSet**
9. Get the edited and encrypted message that belongs to the patient from the **ResultSet** then assign to the variable **editedencrytedMessage**
10. Send the encrypted messages to the respective patient using the http servlet response

Figure 6-17: *Edited and Encrypted Message Sender Algorithm*

- ***Patient Privacy Protection Policy Listener and Storing Service Component***: this is implemented by a java servlet program that uses http protocol in order to get policies from the remote patients' mobile terminal and store the policy in the database. Algorithm 6.17 contains a set of instructions that implements the patient privacy protection policy listener and storing component as shown in Figure 6-18.

Algorithm 6.17

Input - Task identification **id**
- Name of the end user **Euser**
- Phone number of the patient **patientphoneNo**
- Phone number of the respective physician **physicianphoneNo**
- name of disease (information) **disease**
- patient privacy protection code **encryptedidentity**
- Patient privacy protection **preferenceIndicator**
- **age** attribute
- **sex** attribute
- **country** attribute
- **city** attribute
- **kebele** attribute
- data of expire for access right **expireDate**

Output - **Policy** storing notification

1. Create the DataInputStream **in** for the http servlet request sent from the patient mobile terminal
2. From the data input stream object **in** get all the parameter to set the policy then assign to the variables task identification **id** for policy setting, user of medical information **Euser**, patient phone number who is setting the policy **patientphoneNo**, phone number of the patient's physician **physicianphoneNo** , the kind of disease on which to set access control **disease**, code for identity protection **encryptedidentity**, preference indicator of privacy protection **preferenceIndicator**, age of patient **age**, sex of patient **sex** , country of patient **country** of patient, city of patient **city**, kebele of patient **kebele** and expire date of access right given to the end user **expireDate**.
3. Declare the database as **health** , user as **root** and the password of the database as **sa**
4. Create a connection object **conn** to the database from the java servlet using the **root**, **sa** and **health**
5. Create a statement object **stm** using the connection object **conn**
6. Write the query to insert all the attributes stated above into the **accessControl** table in the database
7. Then execute the insert query statement using the object **stm** and get the **affected number of record**
8. IF the affected number of record is different from zero, the policy is stored. Then generate a notification and send it back to the patient using the http servlet response.

Figure 6-18: *Patient Privacy Protection Policy Listener and Storing Algorithm*

- ***Medical Information Request Listener Service Component***: this is implemented by a java servlet program that gets requests from the remote physicians' terminal via the http protocol. It is also responsible to identify the kind of request and forward to the policy interpreter and decision maker component. Algorithm 6.18 contains a set of instructions that implements medical information request listener component as shown in Figure 6-19.

Algorithm 6.18

Input - the requester end user **EndUser**
- the phone number of the patient **patientphoneNo**
- the phone number of the respective physician **physicianphoneNo**
- the required resource **KindofDisease**

Output - provide the specific need of the end user to the policy interpreter and decision maker component

1. Create the DataInputStream **in** for the http servlet request sent from the physician mobile terminal
2. From the data input stream object **in** get the end user then assign to the **EndUser**
3. From the data input stream object **in** get the patient phone number then assign to **patientphoneNo**
4. From the data input stream object **in** get the respective physician phone number then assign to **physicianphoneNo**
5. From the data input stream object **in** get the required information **KindofDisease**
6. Declare the database as **health** , user as **root** and the password of the database as **sa**
7. Create a connection object **conn** to the database from the java servlet using the **root, sa** and **health**

Figure 6-19: *Medical Information Request Listener Algorithm*

- ***Patient Privacy Protection Policy Interpreter and Decision Maker Component***: this is responsible to identify the request parameters sent from the previous component and interpret the policies set for that kind of request (Appendix H). Finally decision will be given before information is provided. This component is implemented by the java servlet that uses http protocol in order to provide the medical information according to the decision made. Algorithm 6.19 contains a set

of instructions that implements the patients' privacy protection policy interpreter and decision maker component as shown in Figure 6-20.

Algorithm 6.19

Input - the request specifying parameters

Output - provide decision based on the policy

1. Create a statement object **stm** using the connection object **conn** created at the medical information request listener component
2. Write the select query to get the policy set by the **patient** on the specified **disease** for that particular **end user**
3. Declare a result set **ResultSet** to hold the returned policy from the **accesscontrol** table in the database
4. Execute the select query statement using the object **stm** and assign the row returned to the **ResultSet**
5. From the **ResultSet** get the values assigned to each attribute
6. IF an attribute is assigned to 1, that attribute is sharable
7. ELSE that attribute is non-sharable
8. IF the **privacy indicator** is 1 when the policy is set, the identity indicators such as name, Telephone number, house number will be protected by the given codes

Figure 6-20: *Patient Privacy Protection Policy Interpreter and Decision Maker Algorithm*

- ***Medical Information Provider Service Component***: this is implemented using the same servlet java program of the previous component in order to send the medical information provided according the decision made. Algorithm 6.20 contains a set of instructions that implements the medical information provider component as shown in Figure 6-21.

Algorithm 6.20

Input - the decisions given by the policy interpreter

Output - provide medical information based on the decision

1. Declare the database as **health** , user as **root** and the password of the database as **sa**
2. Create a connection object **conn** to the database from the java servlet using the **root, sa** and **health**
3. Create a statement object **stm** using the connection object **conn**
4. Write the select query to fetch the medical information of the patient having a phone number **patientphoneNo** and the information related to the disease **KindofDisease** and other details of the patient
5. Declare a result set **ResultSet** to hold the returned medical information from the **medicalhistory** and **patientprofile** tables
6. Execute the select query statement using the object **stm** and assign the row returned to the **ResultSet**
7. Get the detail medical information that belongs to the patient from the **ResultSet**
8. Then send the allowed medical information to the respective physician mobile phone using the http servlet response
9. Then that information will be provided to the End User

Figure 6-21: *Medical Information Provider Algorithm*

- ***Database Component***: this is a MySQL database server in order to store the necessary information for the system (Appendix F, G, H, I).

6.2.3 Components of the Physician Module

- ***Encrypted Secret Key Listener and Decryption service Component***: this is implemented by a J2ME thread that gets the secret key from the remote server and decrypts it in order to get the original key for message encryption/decryption. The key is decrypted by ECC using the private key of the physician. Then the 64 bit secret key is going to be used for blowfish encryption and decryption code that are implemented at the physicians' mobile terminals. Algorithm 6.21 contains a set of instructions that implements the encrypted secret key listener and decryption component as shown in Figure 6-22.

Algorithm 6.21

Input - Private key of the respective physician **Pri**
- Phone number of the patient **patientphoneNo**
- Phone number of the respective physician **physicianphoneNo**
- import the package **org.bouncycastle.crypto**

Output - The secret key **K** in bytes

1. Create an instance **c** of an HttpConnection to the remote server using the url address of the web application that listens the parameters will be sent ;
   HttpConnection c = Connector.open (http://localhost:8080/sendMessage/ secretkeysender)
2. Set the request property of the instance **c**
3. Set the request method of the instance **c** to be a POST httpConnection
4. To send the request and get the secret key from the remote server
   a. **os**.writeUTF(**patientphoneNo**)
   b. **os**.writeUTF(**physicianphoneNo**)
   c. Close the object **os**
   d. Create an instance of a DataInputStream **is** using the http connection instance **c** ;
      DataInputStream **is** = **c**.openDataInputStream()
   e. create a stringBuffer **sb** to accumulate the stream data of the secret key;
      stringBuffer **sb =** new stringBuffer()
   f. declare an integer **ch** that reads each character in the stream; **ch = is**.read()
5. WHILE **ch** is different from -1 (end of the data stream)
   a. Change **ch** (the decimal equivalent of a character) to the equivalent character
   b. Then append the character to the **sb** string buffer ; **sb.append**(char(ch))
6. End of WHILE
7. Then convert the buffered stream data in to string then decode it in base64 then store it in the variable **enckey**
8. Then decrypt the key **enckey** using **ECC** using the private key of the physician as: **K = ECC** (**enckey, Pri**)

Figure 6-22: *Encrypted Secret Key Listener and Decryption Algorithm*

- ***Encrypted Health Status Message Listener Service Component***: this is responsible to get health status messages of patients of a particular physician via http communication protocol from the remote server. It is implemented by a J2ME thread in order to send the required parameters to the remote java servlet then get the health status messages. Algorithm 6.22 contains a set of instructions that implements the encrypted health status message listener component as shown in Figure 6-23.

Algorithm 6.22

Input - the phone number of the physician **physicianphoneNo**

Output - **EncryptedMessage (**in byte array**)** from patients

1. Create an instance **c** of an HttpConnection to the remote server using the url address of the web application that listens the parameters will be sent ;
   HttpConnection c = Connector.open
   (http://localhost:8080/sendMessage/message)
2. Set the request property of the instance **c**
3. Set the request method of the instance **c** to be a POST httpConnection
4. To get the **EncryptedMessage** sent from the patients of this physician from the remote server
   a. Create an instance of a DataOutputStream for the httpConnection instance **c** as;
      DataOutputStream **os** = **c**.openDataOutputStream()
   b. Then send the phone number of the physician
      I **os**.writeUTF (**physicianphoneNo**)
      II Close the object **os**
   c. Create an instance of a DataInputStream **is** using the http connection instance **c** ;
      DataInputStream **is** = **c**.openDataInputStream()
   d. create a stringBuffer **sb** to accumulate the stream data of the incoming message; stringBuffer **sb =** new stringBuffer()
   e. declare an integer **ch** that reads each character in the stream; **ch = is**.read()
5. WHILE **ch** is different from -1 (end of the data stream)
   a. Change **ch** (the decimal equivalent of a character) to the equivalent character
   b. Then append the character to the **sb** string buffer ;
      sb.append(char(ch))
6. End of WHILE

```
7. Declare a String array SMS to hold the message in the string
   buffer
8. For the number of N messages assign the array starting from
   i=0 as:  SMS [i] = sb.toString() until the end of the
   messages
9. The SMS data is a string data that was converted from base64
   encoding which is made for storing purpose
10. So, convert each SMS back to base64;
     a. byte[] base64SMS = Base64.decode (SMS.getBytes (UTF-8))
     b. assign to the EncryptedMessage; EncryptedMessage =
        base64SMS
     c. then send EncryptedMessage  to the decryption component
```

Figure 6-23: *Encrypted Health Status Message Listener Algorithm*

- ***Message Decryption Service Component***: this is a blowfish encryption code that is implemented in the same J2ME thread of the above component in order to decrypt the encrypted message (Appendix B). The implementation of this component is a replica of the Algorithm 6.7 as shown in Figure 6-8.
- ***Message Display Service Component***: the J2ME MIDlet classes are used for the development of graphical user interface (GUI) on which to display the messages. The implementation of this component is a replica of the Algorithm 6.8 as shown in Figure 6-9.
- ***Message Editor and Encryption Service Component***: the health status messages displayed on the previous component are going to be corrected by the physician and sent to the next component. This is implemented by a J2ME thread which gets the messages edited for each patient of a physician and encrypts them before sent to the remote server (Appendix B). The implementation of this component is a replica of the Algorithm 6.4 as shown in Figure 6-5.
- ***Edited and Encrypted Message Sender Service Component***: this is implemented in the same J2ME thread of the previous component. It is responsible to send the secured messages to the remote server via an http communication protocol. Algorithm 6.23 contains a set of instructions that implements the edited and encrypted message sender component as shown in Figure 6-24.

Algorithm 6.23

Input - the encrypted message **EditedandEncryptedSMS**(bytes array)
- the phone number of the patient **patientphoneNo**
- the phone number of the respective physician **physicianphoneNo**
- import **org.bouncycastle.util.encoders.Base64** to have encoding function

Output - Message **delivery notification**

1. Create an instance **c** of an HttpConnection to the remote server using the url address of the web application that listens the parameters will be sent ;
   HttpConnection c = Connector.open (http://localhost:8080/sendMessage/message1)
2. Set the request property of the instance **c**
3. Set the request method of the instance **c** to be a POST httpConnection
4. Create an instance of a DataOutputStream for the httpConnection instance **c** as;
   DataOutputStream **os** = **c**.openDataOutputStream()
5. Convert the **EditedandEncryptedSMS** (in byte array) to its base64 equivalent before sent to the remote server to be stored; byte[] **EncB64SMS = Base64.encode (EditedandEncryptedSMS)**
6. Then convert the **base64** equivalent to **string** before sending the message; String **encryptedData** = new String (**EncB64SMS**)
7. Then send the message, phone number of the patient and the respective physician
   a. **os**.writeUTF (**encryptedData**)
   b. **os**.writeUTF (**patientphoneNo**)
   c. **os**.writeUTF (**physicianphoneNo**)
   d. Close the object **os**
8. To get the message delivery notification
   a. create an instance of a DataInputStream **is** using the http connection instance **c** ;
      DataInputStream **is** = **c**.openDataInputStream()
   b. create a stringBuffer **sb** to accumulate the stream data of the notification;
      stringBuffer **sb =** new stringBuffer()
   c. declare an integer **ch** that reads each character in the stream; **ch = is**.read()
9. WHILE **ch** is different from -1 (end of the data stream)
   a. Change **ch** (the decimal equivalent of a character) to the equivalent character
   b. Then append the character to the **sb** string buffer ;

```
            sb.append(char(ch))
   10. End of WHILE
   11. Display the notification sent from the remote server on the
       physician mobile terminal
```

Figure 6-24: *Edited and Encrypted Message Sender Algorithm*

- ***Medical Information Request Sender Service Component***: this component is responsible to send access request for medical information initiated by physicians to the remote server. This is implemented by a J2ME thread using an http protocol. Algorithm 6.24 contains a set of instructions that implements the medical information request sender component as shown in Figure 6-25.

Algorithm 6.24

Input - the requester end user **EndUser**
- the phone number of the patient **patientphoneNo**
- the phone number of the respective physician **physicianphoneNo**
- the required resource **KindofDisease**

Output - the allowed **medical information** will be received by the **medical information listener component**

1. Create an instance **c** of an HttpConnection to the remote server using the url address of the web application that listens the parameters will be sent ;
   HttpConnection c = Connector.open
   (http://localhost:8080/sendMessage/ message)
2. Set the request property of the instance **c**
3. Set the request method of the instance **c** to be a POST httpConnection
4. To send the request and get the allowed information
   a. **os**.writeUTF(**EndUser**)
   b. **os**.writeUTF(**patientphoneNo**)
   c. **os**.writeUTF(**physicianphoneNo**)
   d. **os**.writeUTF(**KindofDisease**)
   e. Close the object **os**

Figure 6-25: *Medical Information Request Sender Algorithm*

- ***Medical Information Listener Service Component***: this is implemented by a J2ME thread using an http protocol in order to fetch the allowed medical information from the remote server at the physicians' mobile terminal. Algorithm 6.25 contains a set

of instructions that implements the medical information listener component as shown in Figure 6-26.

Algorithm 6.25

Input - the request is sent by the **medical information request sender component**

Output - the allowed **medical information** of the selected patient according to the policy set by the patient on that particular disease

1. To get the medical information according to policy in the database
   a. create an instance of a DataInputStream **is** using the http connection instance **c** ;
      DataInputStream **is** = **c.**openDataInputStream()
   b. create a stringBuffer **sb** to accumulate the stream of the information that consists a number of attributes of data concatenated together;
      stringBuffer **sb =** new stringBuffer()
   c. declare an integer **ch** that reads each character in the stream; **ch = is.**read()
2. WHILE **ch** is different from -1 (end of the data stream)
   a. Change **ch** (the decimal equivalent of a character) to the equivalent character
   b. IF
      - the character is different from the indicator character of an attribute of data, then continue appending the character to the **sb** string buffer ; **sb.append**(char(ch))
   c. ELSE
      - Display the attribute from the **sb** string buffer then create the a new string Buffer for the next attribute of the information stream and continue appending the next character
3. End of WHILE
4. The attributes of the medical information will be displayed on the mobile of the physician for sharing with the end user

Figure 6-26: *Medical Information Listener Algorithm*

6.3 Scenarios of privacy concerned patients

In this subsection the scenarios are prepared to show how the overall system works. The pervasive healthcare sub-system and the privacy protection manager sub-system are going to be tested based on the scenarios. A total of 14 patients and two doctors are selected to test the system towards privacy protection of the patients. The reason for having two doctors who have different groups of patients is to show how the system protects patients' medical information from access by other physicians who did not give any treatment to them. The scenarios are used to show how patients' privacy is protected and their medical information is shared with others in such a way that helps the users of medical information.

1. Almaz Tefera is an HIV patient and she has seen by Dr. Tenna Abate. She sends her CD4 count to her doctor every 6 month remotely. Beyene Kebede, the husband of Almaz Tefera is also an HIV patient and he has also seen by Dr. Abebe Ytna. He also sends his CD4 count to his doctor every 6 month remotely.
2. Dr. Tenna Abate expects to see CD4 status of his HIV patients. Dr. Abebe Ytna also expects to see CD4 status of his HIV patients. The CD4 status is a message generated at the patient terminal in a way that helps physicians what the count indicates for. Dr. Tenna Abate and Dr. Abebe Ytna has opened their terminal and read the CD4 status messages. The two doctors has edited the message and sent it back to their patients.
3. Later on Almaz and Beyene has opened their terminal to check the message has come from their doctors.
4. Dawet Henok is a patient of Dr. Tenna Abate and he needs to classify his medical resource in what way should it be shared with the users of medical records. He needs his medical information related to the disease HIV must be provided to medical researchers by hiding the identity indicator attributes. But he does not worry about privacy for the case of his payer organization if that needs medical evidence in order to take care of the bill. For the case of medical researchers, he allowed the attributes; name of the disease, treatment taken, date of birth, sex, country and city only and his name and house number in coded form.

5. Dr. Tenna Abate is asked to provide medical information of his patients by different users. A medical researcher has requested to have medical details of HIV patients for data analysis of the researcher's work. Payer organization has asked to know the current health problem of Dawet Henok and his expenses for evidence purpose.
6. Dr. Tenna Abate wants to see the medical history of Dawet Henok to know the diseases and treatments he has taken so far.
7. Dr. Tenna Abate wants detail medical information of all his patients for data analysis. The data should be exported as csv file to be opened on excel.

6.4 Demonstration

In this subsection, the prototype implementation will be demonstrated based on the scenarios stated in the previous subsection. The scenarios are implemented as follows:

Scenario 1: in the first scenario, Almaz and Beyene are going to send their CD4 count. According the prototype implementation, the patients' mobile generates a maximum of three random integer numbers in order to have the highest probability of getting the CD4 less than or equals to 350 cells/mm^3 during testing. The code that emulates the Mixi z device is available at Appendix J. Then if one of the three generated random integers is less than or equals to 350 cells/mm^3, accordingly health status message is going to be generated. Finally the message is going to be encrypted using blowfish encryption code then sent to the remote server via http connection (Appendix K). The graphical user interface of this scenario is shown in Figure 6-27.

Figure 6-27: *Task selection to see messages and/or to run Moxi*

Scenario 2: in the second scenario, Dr. Tenna and Dr. Abebe want to see their patients' health status. In the implementation, the messages (SMS) are displayed on their GUI based on the identity of the physicians sent to the server (Appendix E). Accordingly they will edit the messages by selecting the name of the patients from the list. The messages already edited are going to be sent to the server to be available for the respective patients. Part of the

implementation is found at Appendix L. The graphical user interface at the physicians' mobile terminal is as shown in Figure 6-28. As shown in Figure 6-29, Dr. Tenna is editing Almaz's message. Similarly Dr. Abebe is editing Beyene's message (Appendix Q).

Figure 6-28: *Task selection to see patients' CD4 status*

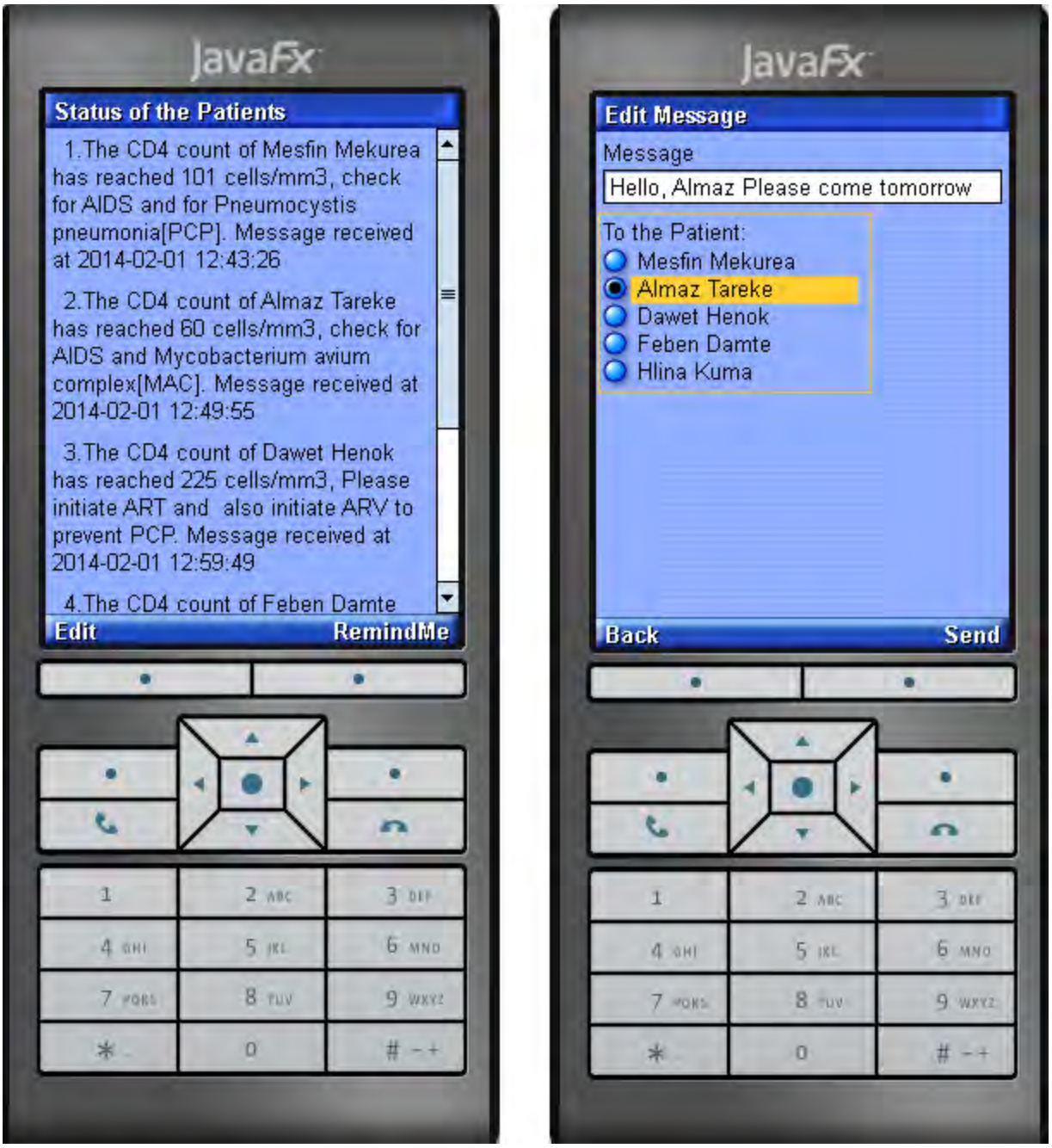


Figure 6-29: *Messages arrived at Dr. Tenna's smartphone*

Scenario 3: in the third scenario, Almaz and Beyene want to see the messages sent from their doctors. This is implemented by a J2ME thread which sends identity of the owner of the mobile via http connection to the server in order to get the messages that belongs to that particular patient back (Appendix D). At the web server side servlets are responsible to get the messages and send to that patient. The message sent to Almaz and Beyene are available at Appendix R.

Scenario 4: in scenario 4, Dawet wants to set access control on his resource (HIV related information). In the implementation case, there are integer variables that are initially assigned to 0 then according to the selected attributes on the graphical user interface of the patient they are going to be assigned to 1 (Appendix H). If a snippet of the implementation code is required (Appendix M). The variables represent each attribute found in the medical information and give a meaning as the attributes are sharable (1) or not. Those attributes are sent to the server and saved in the database to be used as governing policies during information sharing. Dawet has protected his privacy for the case of medical research in his selected resource (HIV) as shown in Figure 6-30. He has also set expire date of access and the allowed attributes for researchers in that information as shown in Figure 6-31. To protect his privacy, his identity will be coded (Appendix C). Similarly he has set an access control policy for his payer organization. The graphical user interface is available at Appendix S and Appendix T. For the case of his payer organization, he allowed to get any information related to that resource (HIV).

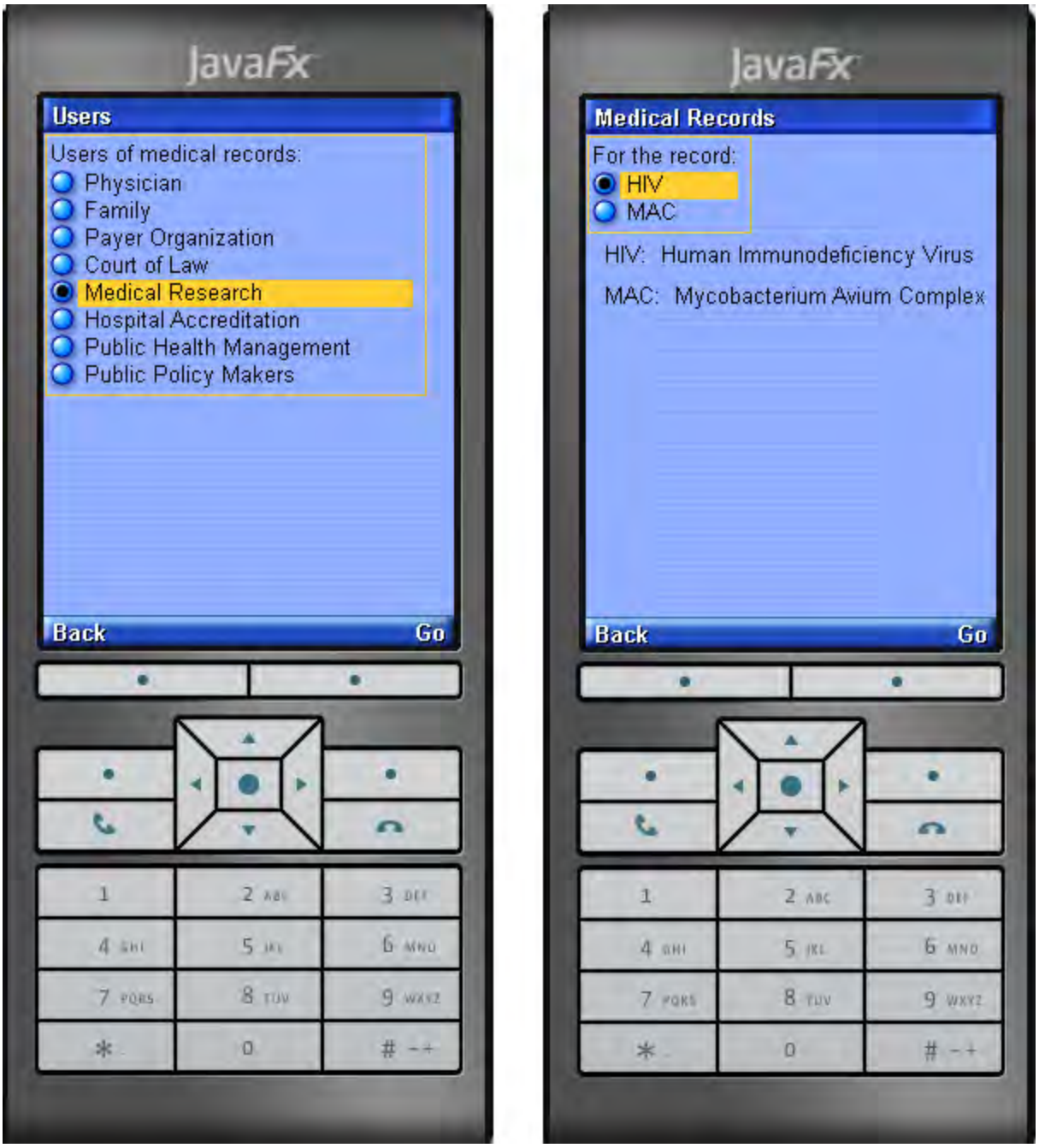


Figure 6-30: *User and resource selection to set policy for medical research*

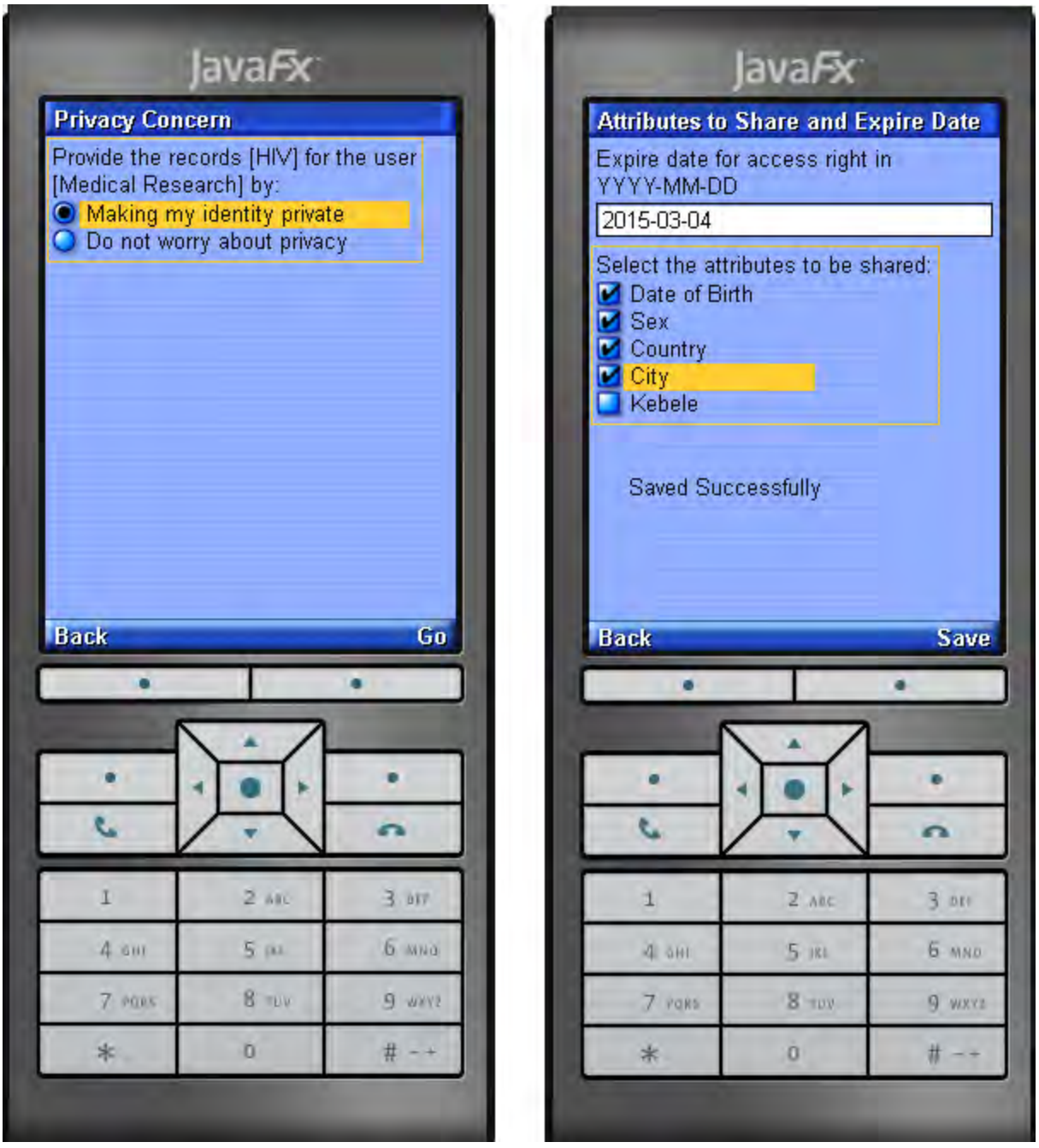


Figure 6-31: *Setting policy on HIV for the use of medical research*

Scenario 5: in the fifth scenario, Dr. Tenna has two requests on medical information of his patients from a medical researcher and payer organization of Dawet. It is implemented by a J2ME thread which takes the parameters according to the request of the physician then sends it to the remote server for decision making based on the policies. A snippet of the implementation code is available at Appendix N. The parameters are: requester (information user), the

information required and of which patient. Finally, based on the policies the allowed information will be sent to the physician terminal. The physician has specified that he needs to share medical information with a medical researcher. Then, he specified the user who needs the information and in what kind of disease is the user interested as shown in Figure 6-32. Finally, the physician selected the particular patient of which the detail medical information is required by the user as shown in Figure 6-33. Similarly for the case of Dawet's payer organization, it is available at Appendix U and Appendix V.

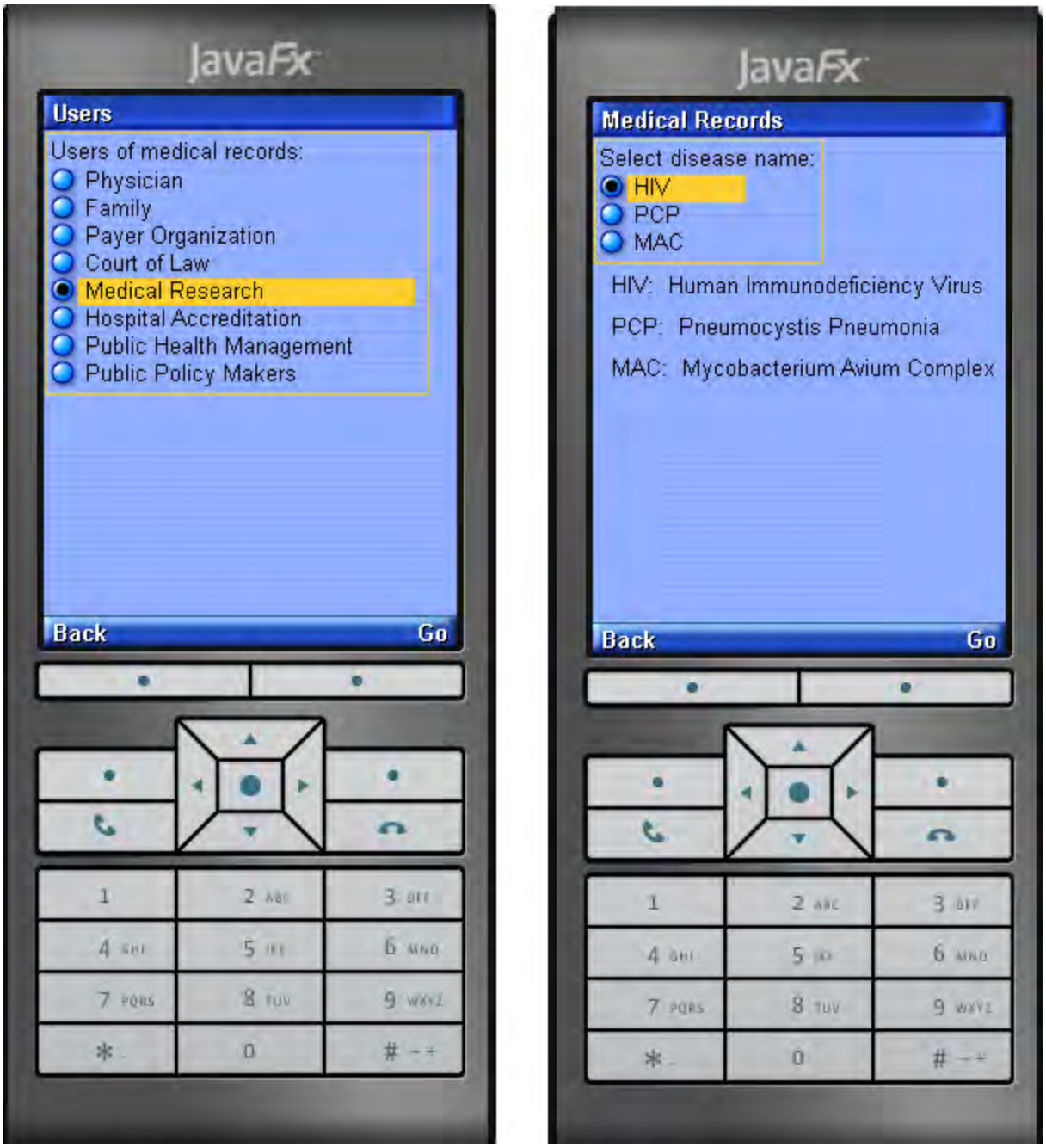


Figure 6-32: *The medical researcher's need of resource*

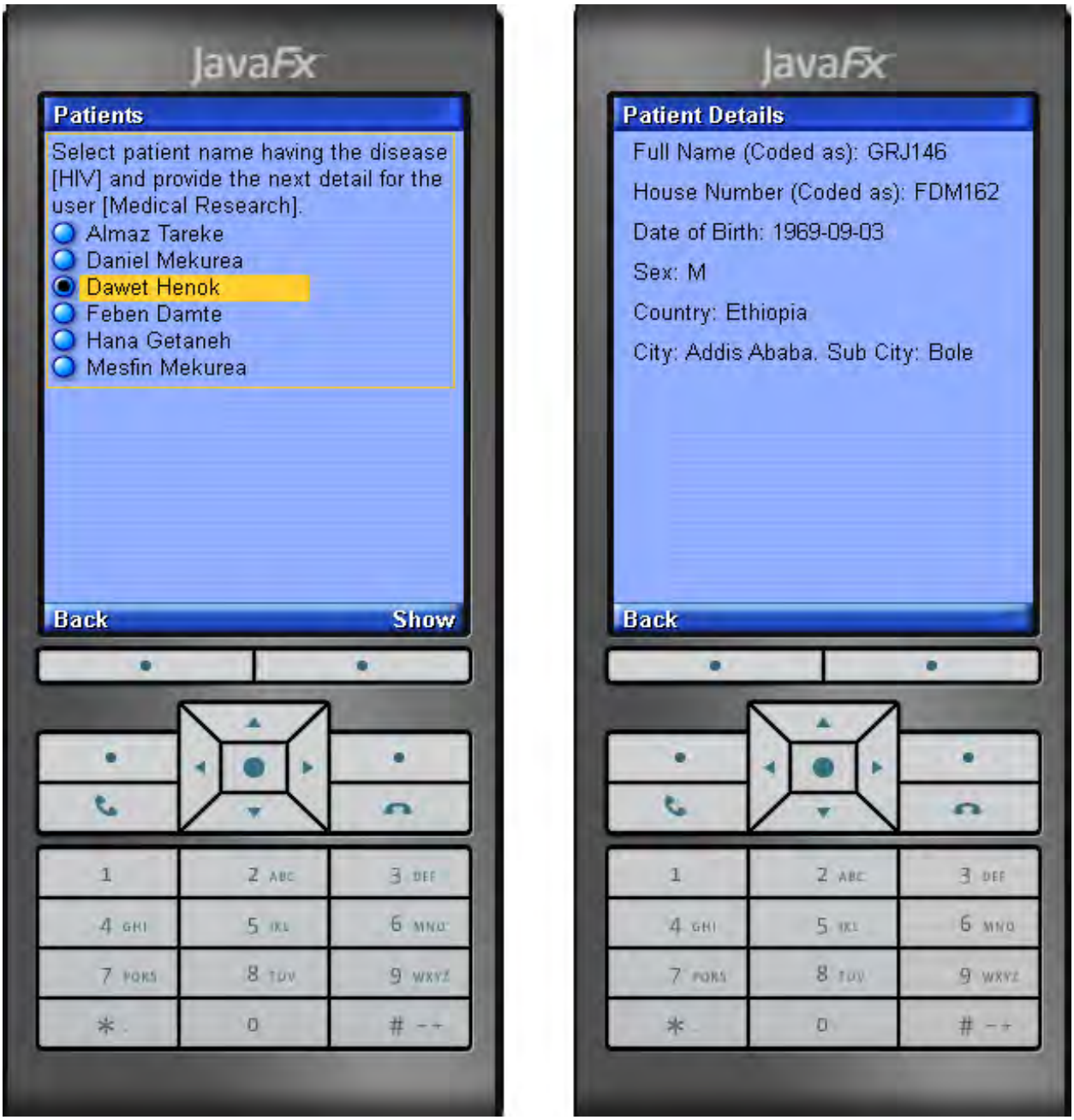


Figure 6-33: *Detail information of Dawet Henok that is allowed for medical researchers*

Scenario 6: in the sixth scenario, Dr. Tenna wants to see Dawet's medical history. When the Midlet starts a J2ME thread is implemented which sends phone number of Dr. Tenna to the remote server in order to get the names of Dr. Tenna's patients (Appendix I) and display on his phone. Another J2ME thread is implemented for the show command that sends phone number of Dr. Tenna and the selected patient name to the remote server in order to get the medical history

of the patient (Dawet). A snippet of code for the implementation is found at Appendix O. The graphical user interface for this scenario is shown in Figure 6-34.

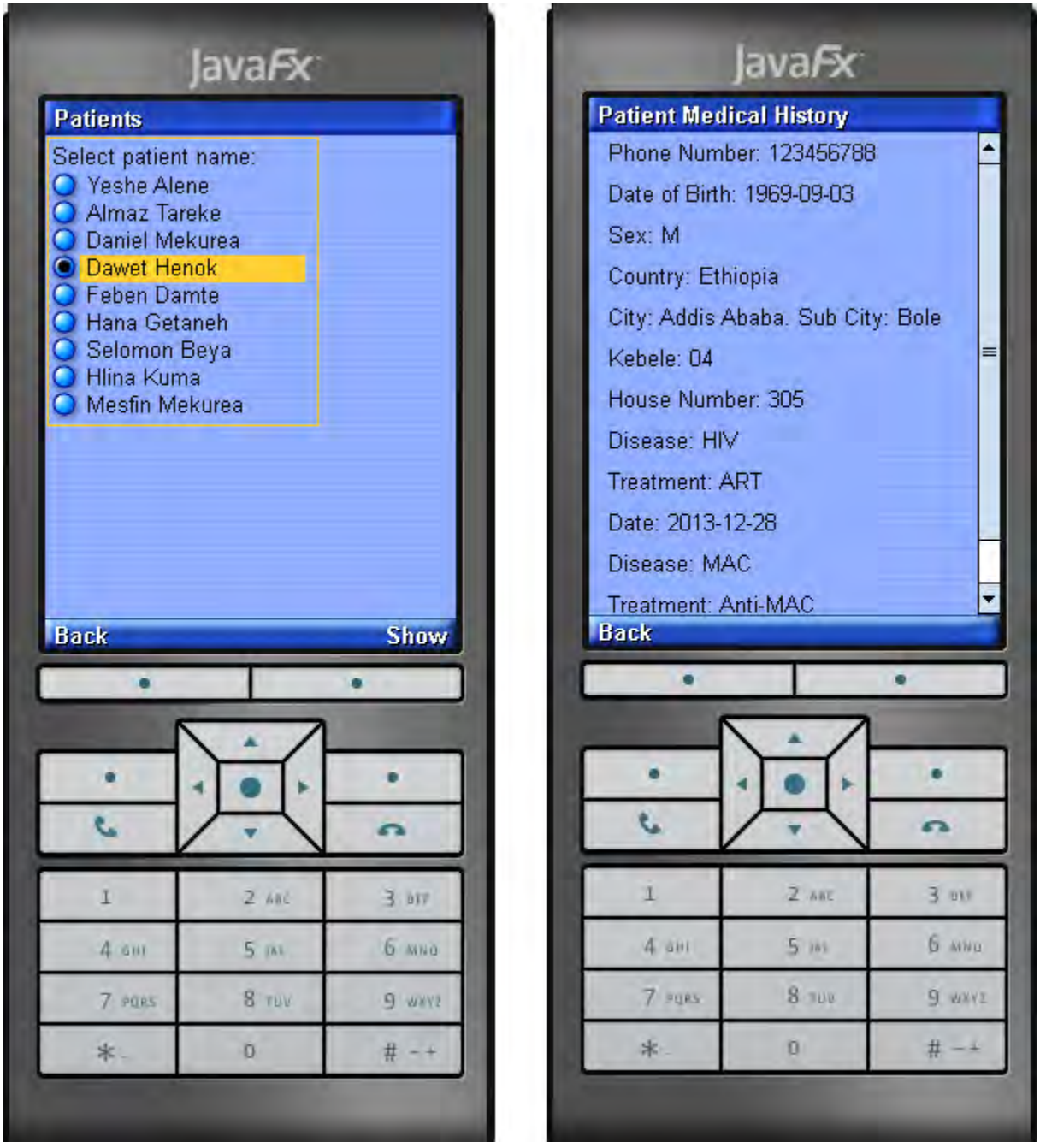


Figure 6-34: *Medical history of Dawet Henok*

Scenario 7: in the seventh scenario, Dr. Tenna wants to get detail medical information of all his patients for data analysis. It is implemented by a J2ME thread that sends the parameters: his phone number and the name for the csv file that will be generated and stored at the remote server

in his active directory. Once the file is created the servlet program sends a confirmation back to the doctor mobile terminal. If code of the implementation is required, a snippet of the code is found at Appendix P. Figure 6-35 shows the graphical user interface to export the data. The exported data is available at Appendix W.

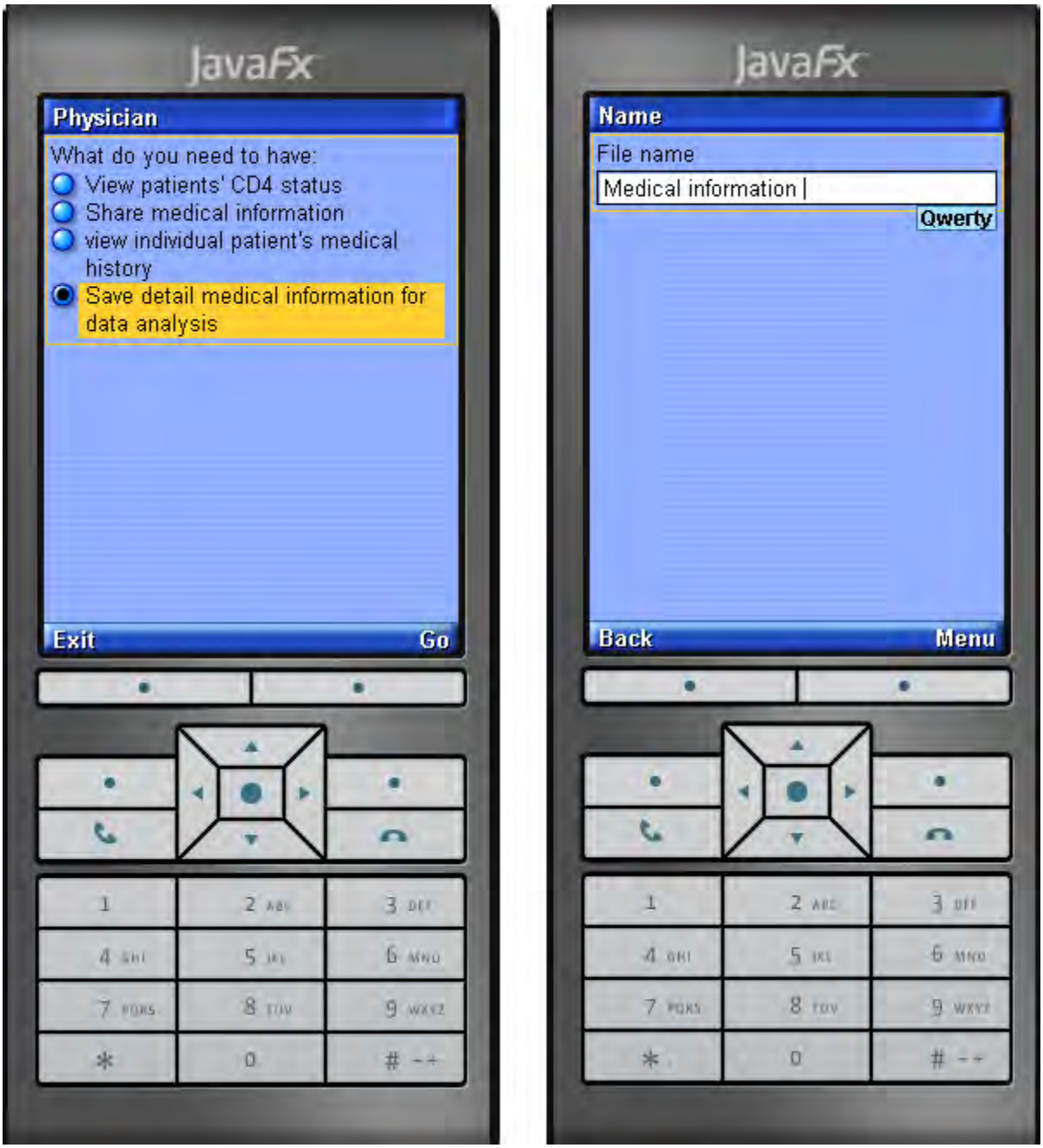


Figure 6-35: *File name for data export*

Chapter 7 : Conclusion and Future Directions

7.1 Conclusion

As we have already mentioned in the previous chapter, privacy concern is critical issue both for inpatients and outpatients. The analysis result shows that outpatients are more privacy aware than inpatients since they are the ones who have more contact with the society. Therefore, they fear that they might be discriminated by the society. Thus, medical information of patients should be protected properly though it is required by different bodies. The survey we made shows that the existing system of patients' medical information management is manual and is prone to security and privacy threat. Medical records can be damaged due to disasters without backup and relatively easy for unauthorized access which exposes to patients privacy violation. It is also time consuming to find information from such manual systems. In addition to that physicians protect privacy of patients by coding patients' name or kind of disease accordingly. But in some cases, when medical researchers need medical information of HIV patients, physicians provide the data by coding names of the patients. However, coding names of all the patients' does not mean that all the patients are privacy concerned rather based on their general conclusion. It is also true that, if patients have the opportunity to set privacy protection policy on their medical information and some of them does not worry about privacy, for some users that information may be very important than the information that contains the names of all the patients are coded.

Pervasive systems by their nature add more vulnerability to the patients' privacy. In this work we proposed a novel Privacy Aware Pervasive Healthcare System (PAPHS) that overcomes the problems faced on patients privacy concern in pervasive environment. The system uses six selected techniques and has five components (modules) in order to solve the problem. The system is evaluated based on the scenarios presented in the previous chapter. The PAPHS avoids patients' privacy violations using the privacy management techniques that are explained in the previous chapters. The system is not prone to an unauthorized access as compared to the existing system. Although pervasive healthcare systems are exposed to unauthorized access due to the wireless communication, our system uses encryption techniques in order to protect privacy of patients. The system helps owners of medical information to set access limitation on their medical information based on their exact need. This is advantageous for physicians to make the exact decision while sharing information with others unlike in the existing system. Thus, users of

medical information especially medical researchers get a benefit from non-privacy concerned patients. For example, if two patients are living in the same house number and their privacy protection policy on a particular resource is based on the code of their identity, the system gives the same code for their house number. This helps the analysts to know how many patients of that particular disease are living in that particular house with the achievement of the patients' anonymity and the success of the required data analysis. The system protects privacy violation of patients as compared to the existing system data management. In the database, patients' CD4 status monitoring messages are stored through time. But that data is used only for controlling the communication of the physicians with their patients only. Therefore, the system controls life time of that data by deleting them from the database every 24 hours since the data is bulky and not suitable to be processed by mobile devices (Appendix F). However, the important messages will be kept as history separately in the database. The system is also evaluated by the physicians based on the demonstration made for the scenarios. According to the feedback of the physicians, the patients can be benefited from the system for the two main reasons:

- Due to fear of social discrimination they would be happy if they can get health service remotely using pervasive healthcare technology.
- They also need to give consent on their medical information before any information provision is made.

They have also mentioned that getting consent of patients easily is very important for physicians and other end users of medical information as well. Finally they have stated that the system is promising for future use.

In pervasive computing environments especially in pervasive healthcare system privacy concern of end users is a critical issue. Patients that use pervasive healthcare systems are highly privacy concerned ones. Among the patients the most privacy concerned are HIV patients as already found from our survey made in Black Lion Hospital and other related papers. In this thesis work, we have developed our own innovative privacy aware pervasive healthcare system (PAPHS) to overcome the privacy concern of HIV patients. Regarding our system the things we have developed are summarized as follows:

- We have emulated a Moxi z sensor in order to have a CD4 count reading. Then the CD4 reading will be sent to the patient's smartphone via Bluetooth communication.

We have checked the CD4 count whether it is in the range of abnormal status or not. If it is not normal, accordingly message (SMS) will be generated then encrypted. Finally, the encrypted message will be sent to the remote server.

- We have developed java servlet programs that get some parameters from the physicians' and the patients' mobile terminals and send some information from the database accordingly.
- Patients' privacy protection policies are set by the owners of medical information that will be sent and stored in the database which is found at the remote server.
- The servlets found in the web server are responsible to decide which information to provide for whom.
- We developed a generic privacy protection architecture that can be applicable for other pervasive healthcare systems as well.

In order to verify the concept mentioned in our proposed system, we have implemented the prototype of the major components of the system. We have demonstrated the overall system using seven different scenarios that raises different situations on privacy concern of patients. We have evaluated our system based on the feedbacks given by the physicians when the system is compared with the existing manual system. We have also evaluated the system by comparing with other privacy aware pervasive environments. The privacy protection techniques used on such systems are specific to the selected pervasive system. But the privacy protection techniques used in our proposed pervasive healthcare system are applicable on other pervasive healthcare systems with some additional techniques if necessary. It is also true that unlike other privacy aware systems, focus of our system is beyond others done. Their focus is on balancing the privacy concern of the users of the pervasive system while providing information to it and the amount of information required by the system in order to improve its efficiency. But Our system also balances the privacy concern of patients who use pervasive healthcare system and other users of medical information who may not have direct access on the pervasive environment rather they use the stored medical information. Finally, we have concluded that our privacy protected healthcare system improves the acceptance of pervasive healthcare systems in the healthcare industry by removing the privacy concern of patients in the pervasive healthcare environment.

7.2 Future Directions

Although we have developed a promising privacy aware pervasive healthcare system, we believe that our system should be improved in some aspects. For instance, there are cases where controlling activities of the patients can be important. In such cases video cameras are required and some other additional techniques will be required to protect privacy of the patients. It is also important to improve the system in order to allow patients to grant different access right for the individual members of their family. Only prototype of the system is implemented due to budget limitation. Therefore, the system can be implemented in the real environment with the major requirements of Moxi z, smartphone and web server in order to make it usable system.

References

[1] M. Weiser, The computer for the 21st century. ACM SIGMOBILE Mobile Computing and Communications Review, 1999, pp. 3, 5.

[2] G. Borriello, V. Stanford, C. Narayanaswami, W. Menning, Pervasive computing in healthcare. IEEE Pervasive Computing, 2007.

[3] F. Adams, editor, Genuine Works of Hippocrates. Krieger Publishing, 1972.

[4] I. Korhonen, J.E. Bardram, Guest Editorial Introduction to the Special Section on Pervasive Healthcare. IEEE Transactions on Information Technology in Biomedicine, 2004.

[5] M. Tentori, R. Favela, M.D. Rodriguez, "Privacy-aware Autonomous Agents for Pervasive Healthcare," in IEEE Intelligent Systems, Nov-Dec 2006, pp. 55-62.

[6] M. Weiser, "Some Computer Science Problems in Ubiquitous Computing," Communications of the ACM, Vol. 36, No. 7, Jul 1993, pp. 75-84.

[7] J. Al-Muhtadi, R. Campbell, A. Kapadia, M.D. Mickunas, Y. Seung, "Routing through the Mist: Privacy Preserving Communication in Ubiquitous Computing Environments," 22nd International Conference on Distributed Computing Systems, 2002, pp. 74-83.

[8] V. Bellotti, A. Sellen, "Design for Privacy in Ubiquitous Computing Environments," 3rd European Conference on Computer Supported Cooperative Work, 1993, pp. 77-92.

[9] M. Satyanarayanan, "Privacy: The Achilles Heel of Pervasive Computing?," IEEE Pervasive Computing, Jan-Mar 2003, 2(1), pp. 2-3.

[10] A.R. Jacobs, G.D. Abowd, "A Framework for Comparing Perspectives on Privacy and Pervasive Technologies," IEEE Pervasive Computing, Oct-Dec 2003, 2(4), pp. 78-84.

[11] R.S. Cardoso, V. Issarny, "Architecting Pervasive Computing Systems for Privacy: A Survey," IEEE/IFIP Conference on Software Architecture, Jan 2007, pp. 26-29.

[12] J.I. Hong, J.A. Landay, "An Architecture for Privacy-Sensitive Ubiquitous Computing," 2nd international Conference on Mobile Systems, Applications, and Services, Jun 2004, pp. 177-189.

[13] M.J. Culnan, "Protecting privacy online: is self-regulation working?," Journal of Public Policy Market, 2006, 19(1), pp. 20-26.

[14] R. Babbitt, J. Wong, C. Chang, "Towards the Modeling of Personal Privacy in Ubiquitous Computing Environments," 31st Annual International Computer Software and Applications Conference, Jul 2007, pp. 695-699.

[15] J. Cas, "Privacy in Pervasive Computing Environments – a Contradiction in Terms?," Technology and Society Magazine, Spring 2005, 24(1), pp. 24-33.

[16] M. Langheinrich, "Personal Privacy in Ubiquitous Computing – Tools and System Support," PhD thesis, ETH Zurich, Switzerland, May 2005.

[17] S. Lederer, J.I. Hong, A.K. Dey, J.A. Landay, "Personal Privacy through Understanding and Action: Five Pitfalls for Designers," Personal and Ubiquitous Computing, Nov 2004, 8(6), pp. 440-454.

[18] M.S. Ackerman, "Privacy in Pervasive Environments: Next Generation Labeling Protocols," Personal and Ubiquitous Computing, Nov 2004, 8(6), pp. 430-439.

[19] C. Brodie, C.-M. Karat, J. Karat, J. Feng, "Usable Security and Privacy: a Case Study of Developing Privacy Management Tools," Symposium on Usable Privacy and Security, 2005, pp. 35-43, 2005.

[20] Computing Research Association, "Four Grand Challenges in Trustworthy Computing," Nov 2003, http://www.cra.org/reports/trustworthy.computing.pdf, retrieved on September 3, 2013.

[21] D. Anthony, T. Henderson, D. Kotz, "Privacy in Location-Aware Computing Environments," IEEE Pervasive Computing, Oct-Dec, 2007, 6(4), pp. 64-72.

[22] G.R. Hayes, E.S. Poole, G. Iachello, S.N. Patel, A. Grimes, G.D. Abowd, K.N. Truong, "Physical, Social, and Experiential Knowledge in Pervasive Computing Environments," IEEE Pervasive Computing, Oct-Dec 2007, 6(4), pp. 56-63.

[23] V. Kostakos, E. O'Neill, A. Penn, "Designing Urban Pervasive Systems," Computer, Sep 2006, 39(9), pp. 52-59.

[24] D. Halperin, T.S. Heydt-Benjamin, K. Fu, T. Kohno, W.H. Maisel, "Security and Privacy for Implantable Medical Devices," IEEE Pervasive Computing, Jan/Mar 2008, 7(1), p. 30-39.

[25] G. Pallapa, N. Roy, S. Das, "Precision: Privacy Enhanced Context-Aware Information Fusion in Ubiquitous Healthcare," 1st International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments, 2007, sepcase, p. 10.

[26] S. Warren, L. Brandeis, The right to privacy, Harvard Law Review, 4:193 – 220, 1890.

[27] A.F. Westin, Privacy and Freedom, Atheneum, New York NY, 1967.

[28] A. Mequanint , "Context Aware Pervasive Healthcare System for HIV/AIDS Patients (CAPHS) ,"Tanzania:IST-Africa 2012 Conference & Exhibition, May 2012.

[29] W.E. Mackay, "Spontaneous Interaction in Virtual Multimedia Space: EuroPARC's RAVE system," Imagina, 1992.

[30] S. Balfe, S. Li, J. Zhou, "Pervasive Trusted Computing," 2nd International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, 2006, pp. 88-94.

[31] M. Langheinrich, "A Privacy Awareness System for Ubiquitous Computing Environments," 4th International Conference on Ubiquitous Computing, Lecture Notes in Computer Science, Vol. 2498, pp. 237-245.

[32] K.M. Asif, S.I. Ahamed, N. Talukder, "Avoiding Privacy Violation for Resource Sharing in Ad hoc Networks of Pervasive Computing Environment," 31st Annual International Computer Software and Applications Conference, 2007, pp. 269-274.

[33] S. Lachmund, T. Walter, L. Gomez, L. Bussard, E. Olk, "Context-Aware Access Control Making Access Control Decisions Based on Context Information," 3rd Annual International Conference on Mobile and Ubiquitous Systems, Jul 2006, pp. 1-8.

[34] L. Pareschi, D. Riboni, A. Agostini, C. Bettini, "Composition and Generalization of Context Data for Privacy Preservation," 6th Annual IEEE International Conference on Pervasive Computing and Communications, 2008, pp. 429-433.

[35] L. Pareschi, D. Riboni, C. Bettini, "Protecting Users' Anonymity in Pervasive Computing Environments," 6th Annual IEEE International Conference on Pervasive Computing and Communications, 2008, pp. 11-19.

[36] A.J. Blazic, K. Dolinar; J. Porekar, "Enabling Privacy in Pervasive Computing Using Fusion of Privacy Negotiation, Identity Management and Trust Management Techniques," 1st International Conference on the Digital Society, Jan 2007, pp. 30-35.

[37] E. Papadopoulou, S. McBurney, N. Taylor, M.H. Williams, K. Dolinar, M. Neubauer, "Using User Preferences to Enhance Privacy in Pervasive Systems," 3rd International Conference on Systems, 2008, pp. 271-276.

[38] I. Roussaki, M. Strimpakou, C. Pils, N. Kalatzis, M. Neubauer, C. Hauser, M. Anagnostou, "Privacy-Aware Modelling and Distribution of Context Information in Pervasive Service Provision," 2006 ACS/IEEE International Conference on Pervasive Services, Jun 2006, pp. 150-160.

[39] D.M. Konidala, D.N. Duc, L. Dongman, K. Kwangjo,"A Capability-Based Privacy-Preserving Scheme for Pervasive Computing Environments," 3rd IEEE International Conference on Pervasive Computing and Communications, Mar 2005, pp. 136-140.

[40] U. Hengartner, P. Steenkiste, "Exploiting Hierarchical Identity-Based Encryption for Access Control to Pervasive Computing Information," 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks, Sep 2005, pp. 384-396.

[41] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman, "Role based access control models," IEEE Computer, Feb 1996, 29(2): pp. 38-47.

[42] X. Jiang, J.A. Landay, "Modeling Privacy Control in Context-Aware Systems," IEEE Pervasive Computing, Jul-Sep 2002, 1(3), pp. 59-63.

[43] K. Henricksen, R. Wishart, T. McFadden, J. Indulska, "Extending Context Models for Privacy in Pervasive Computing Environments," 3rd IEEE International Conference on Pervasive Computing and Communications, Mar 2005, pp.20-24.

[44] R. Ramli, N. Zakaria, P. Sumari, Privacy Issues in Pervasive Healthcare Monitoring System: A Review, World Acad. Sci. Eng. Technol, Dec 2010, Vol. 48.

[45] World Medical Association, Medical Ethics Manual, 2nd edition. 2009, http://www.wma.net/en/30publications/30ethicsmanual/pdf/ethics_manual_en.pdf.

[46] World Medical Association, International Code of Medical Ethics, Oct 1983, http://history.nih.gov/research/downloads/ICME.pdf.

[47] Comhairle na nDochtuiri Leighis Medical Council, Guide to Professional Conduct and Ethics for Registered Medical Practitioners, 7th edition. 2009, http://www.medicalcouncil.ie/Registration/Guide-to-Professional-Conduct-and-Behaviour-for-Registered-Medical-Practitioners.pdf.

[48] S. Kurkovsky, O. Rivera, J. Bhalodi, "Classification of Privacy Management Techniques in Pervasive Computing," International Journal of u- and e-Service, Science and Technology, 2008, Vol.1.

[49] A. Appari , M. E. Johnson, "Information security and privacy in healthcare: Current state of research," Proc. Workshop on Information Security and Privacy (WISP), Aug. 2008, http://www.ists.dartmouth.edu/library/416.pdf, retrieved on September 9, 2013.

[50] G. Samson, M. Federico, A. M. Khalid,V. Remi, and K. Jrn,"Secure data storage for mobile data collection systems," In Janusz Kacprzyk, Dominique Laurent, and Richard Chbeir, editors, MEDES'12 - International Conference on Management of Emergent Digital Eco Systems, 2012, pages131-144.

[51] M. Dekker, Security of the Internet, the Froehlich/Kent Encyclopedia of Telecommunications, 1997.

[52] W. Stallings, Network Security Essentials, applications and standards, Prentice Hall, 2000.

[53] W. Stallings, Data and Computer Communications, 5thedition, Prentice Hall, 1998.

[54] C. Andersson, GPRS and 3GWireless Applications, WileyPublishing , 2001.

[55] WAP Forum, WAP1.2.1 and 2.0 Specifications, 2003,http://www.wapforum.org/.

[56] Sun Microsystems, MIDP 2.0 Specification, 2002,http://java.sun.com/products/midp/ .

[57] A.E. Diaa Salama, M.A. Hatem, M.H. Mohie, "Performance evaluation of

symmetric encryption algorithms on power consumption for wireless devices," International Journal of Computer Theory & Engineering, 2009,Vol. 1, No. 4.

[58] G. Kamlesh, S. Sanjay, ECC over RSA for Asymmetric Encryption: A review, International Journal of Computer Science Issues, May 2011,vol.8.

[59] T. Struk, "Elliptic Curve Cryptography as Suitable Solution for Mobile Devices," National University of Ireland, Galway, August 28, 2009.

[60] B. Assefa , "Security Enhanced SMS for Sensitive Applications," Department of Computer Science, AAu (MSc thesis),Nov 2013.

[61] http://www.amednews.com/article/20120820/business/308209965/6/, retrieved on September 9, 2013 at 10:20:24.

[62] Data Protection Acts 1988 and 2003, www.dataprotection.ie, retrieved on November 3, 2013.

[63] National Hospitals Office Code of Practice for Healthcare Records Management (www.hse.ie/eng/Publications/Hospitals/National_Hospitals_Office_Code_ofPractice_for_ Healthcare_Records_Management.html) Guidelines of the Irish College of General Practitioners National General Practice Information Technology Group (www.gpit.ie) , retrieved on November 3, 2013.

[64] Data Protection Acts 1988 and 2003, www.dataprotection.ie; Freedom of Information Act 1997, section 6. www.foi.gov.ie/legislation/freedom-of-information-act-1997-2/part-ii-access-to-records, retrieved on November 3, 2013.

[65] Freedom of Information Act 1997 section 28 (6) Regulations 1999, http://www.foi.gov.ie/regulations/regulation.2006-12-06.4301750217, retrieved on November 3, 2013.

[66] Health Protection Surveillance Centre, www.ndsc.ie/hpsc/NotifiableDiseases/Notification Legislation and Process/, retrieved on November 3, 2013.

[67] http://www.missbdesign.com/clients/TAAAC/about_BlackLionHospital.html ; retrieved on July 16, 2013 at 11:15:13.

[68] https://www.ncbi.nlm.nih.gov/m/pubmed/23670110/?i=2&from=/18228562/related, Retrieved on September 13, 2013 at 09:17:12.

[69] http://www.orflo.com/overview.html, Retrieved on September 13, 2013 at 12:19:21.

[70]http://www.who.int/mediacentre/news/releases/2013/new_hiv_recommendations_20130630/ en /, Retrieved on September 13, 2013.

Appendices

Appendix A: Questionnaire to know healthcare providers' view on patients' privacy concerns

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DEPARTMENT OF COMPUTER SCIENCE

Introduction: This questionnaire is prepared to collect necessary data for a research conducted by an MSc student at the department of computer science, Addis Ababa University. The topic of the research is "privacy aware pervasive healthcare system". We want to find out patients privacy concern on the disclosure of their medical information while using pervasive healthcare systems. Please help our work spending a couple of minutes to fill the questionnaire. All personal information will be kept confidential and only the summary result will be used for analysis.

Thank you for your support.

Year of experience: _____

Please tick the most appropriate options:-

1. What kind of patient is more worried about health information security and privacy from the patients' point of view?
   HIV ☐ Psychiatric Behavior ☐ Sexually transmitted infections (Other than HIV) ☐
   Others ☐ : Please Specify: _____
   _____
2. What are the sources of such privacy concerns of the above patients?
   Social Stigma (feeling of shame, disgrace) ☐
   Fear of Societal Discrimination ☐
   Legal implications ☐
   Insurance issues ☐
   Others ☐ : Please Specify: _____
   _____
3. What security mechanisms do you use in your existing system (Manual/Computer based) to minimize privacy concerns of patients?
   _____
   _____

4. What patient record system do you use to store information of most privacy concerned patients?
   Manual System ☐ : How: _____
   _____
   Computer Based System (Digitized) ☐ : How: _____
   _____

5. What are the privacy concerns of patient records stored in a manual system?
   Un-authorized Access ☐
   Damage due to poor handling or storage ☐
   Disasters ☐
   Others ☐ : Please Specify: _____
   _____

6. Do you think using digitized systems to store patient records of the privacy concerned individuals improve privacy of their information?
   Yes ☐ No ☐ May be ☐ No comment ☐

7. Are there cases that force us to share private medical records of the privacy concerned patients?
   Yes ☐ : Please Specify: _____
   _____
   No ☐

8. As per our general observation, there is an increasing need of patient information sharing between patients and healthcare service providers, healthcare service providers and payer organization. What do you think of sharing such information in the digitized world?
   _____
   _____

9. How much of the privacy concerned patients refuse to reveal important information to physicians due to fear of information disclosure or privacy concern?
   All of them ☐ Most of them ☐ Some of them ☐ None ☐

10. Which age rage of the HIV positive patients are highly concerned about the privacy of their medical records?
    <15 ☐ 15-24 ☐
    25-34 ☐ 35-44 ☐
    45-54 ☐ 55-64 ☐
    65-74 ☐ >74 ☐

11. Apart from diagnosis and treatment, it is clear that patient health records have range of purposes like improving efficiency within healthcare system, to drive public policy development, to advance medical research, to convince payer organizations, public health management and hospital accreditation. However; there is a big privacy concern from the patients' side as there is a security hole during time of sharing of the patients' record

among the stakeholders who use the patients' data for the above purposes. So, how do you balance the above increasing need of patients' data and the privacy concern of patients?

Removing all patient specific Identifiers ☐
Reporting ☐
Coding/Encrypting ☐
Asking patients' consent prior to use of patients' specific data ☐
Others ☐ : Please specify: _____
_____

12. Is there a system that allows patients to approve sharing of their information among physicians and other stakeholders who have a need on the patients' record when there is a need?

Yes ☐ No ☐

13. How much patients, who agree information sharing among physicians, reject the notion of releasing information to third parties including employers and family members?

All of them ☐ Most of them ☐ Some of them ☐ None ☐

14. What is the impact of patients' privacy violation on patients from the medical point of view?

Psychological distress ☐
Loss of trust in healthcare team and system ☐
Reduced adherence to treatment and care ☐
Others ☐ : Please specify: _____
_____

15. What do you think if you are asked to use a pervasive healthcare system that enables you to control your patients remotely through your mobile device?

It simplifies physicians' job ☐
It over exposes patient information ☐
It is more jobs for physicians ☐
Others ☐ : Please specify: _____
_____

16. Such systems are naturally prone to security and privacy risks due to the nature of the technology to be used. In your opinion, how strong should the security and privacy protection be in the system?

Highly secure ☐ Secure ☐ Less secure ☐ Unclassified ☐

17. Are you familiar with security and privacy policy of the country's health system?

Very familiar ☐ Familiar ☐ Less familiar ☐ Unfamiliar ☐

18. Do you respect related rules while treating patients in the hospital?

Always ☐ Most of the time ☐ Sometimes ☐ Never ☐

19. Are you aware of the Ethiopian policy on protection of data, information and devices used to process and store them?

Extremely aware ☐ Very aware ☐ Moderately aware ☐ Slightly aware ☐
Unaware ☐

Thank you for your time.

Appendix B: Code for Blowfish encryption and decryption

```
package SensorPMobile;
import org.bouncycastle.crypto.*;
import org.bouncycastle.crypto.engines.*;
import org.bouncycastle.crypto.modes.*;
import org.bouncycastle.crypto.params.*;

public class Encryptor {

   private BufferedBlockCipher cipher;
   private KeyParameter key;
   public Encryptor( byte[] key ){
 cipher = new PaddedBlockCipher( new CBCBlockCipher( new BlowfishEngine() ) );
      this.key = new KeyParameter( key );  //initialize the secret key
   }
   public Encryptor( String key ){
      //initialize the secret key
      this( key.getBytes() );
   }

   private byte[] callCipher( byte[] data ) throws CryptoException {
      int size = cipher.getOutputSize( data.length );
      byte[] result = new byte[ size ];
      int olen = cipher.processBytes( data, 0,data.length, result, 0 );
      olen += cipher.doFinal( result, olen );
      if( olen < size ){
         byte[] tmp = new byte[olen];
```

```
            System.arraycopy(result, 0, tmp, 0, olen );
            result = tmp;
        }
            return result;
        }
        //encrypt data
        public byte[] encryptString( String data )throws CryptoException {

            if( data == null || data.length() == 0 ){
                return new byte[0];
            }
            else {
                cipher.init( true, key );
                return callCipher( data.getBytes() );
            }
        }
        //decrypt cipher text
        public String decryptString( byte[] data ) throws CryptoException {

            if( data == null || data.length == 0 ){

                return "";

            }
            else{

                cipher.init( false, key );
                return new String( callCipher( data ) ); //convert the byte array into string

            }
        }

    }  //End of Encryptor class
```

Appendix C: Part of the implementation of patients' privacy protection by coding

```
                                        .........
                                        .........
//Give code for identity indicators
        //ASCII for A-Z
        int start = 65;  // ASCII for A
        int end = 90; //ASCII for Z
        Random ARandom = new Random();
        StringBuffer str = new StringBuffer();

        for(int i=0;i<3;i++)
        {
         str.append((char)(start + ARandom.nextInt(Math.abs(end-start))));
        }

        encryptedPname = str.toString(); // the three English alphabets
        int start1=100;
        int end1=200;
        //concatenate the string with the three digit integer
encryptedPname  + = String.valueOf (start1 +  ARandom.nextInt(Math.abs(end1-
start1)));
                                        .........
                                        .........

    }

}//End of class
```

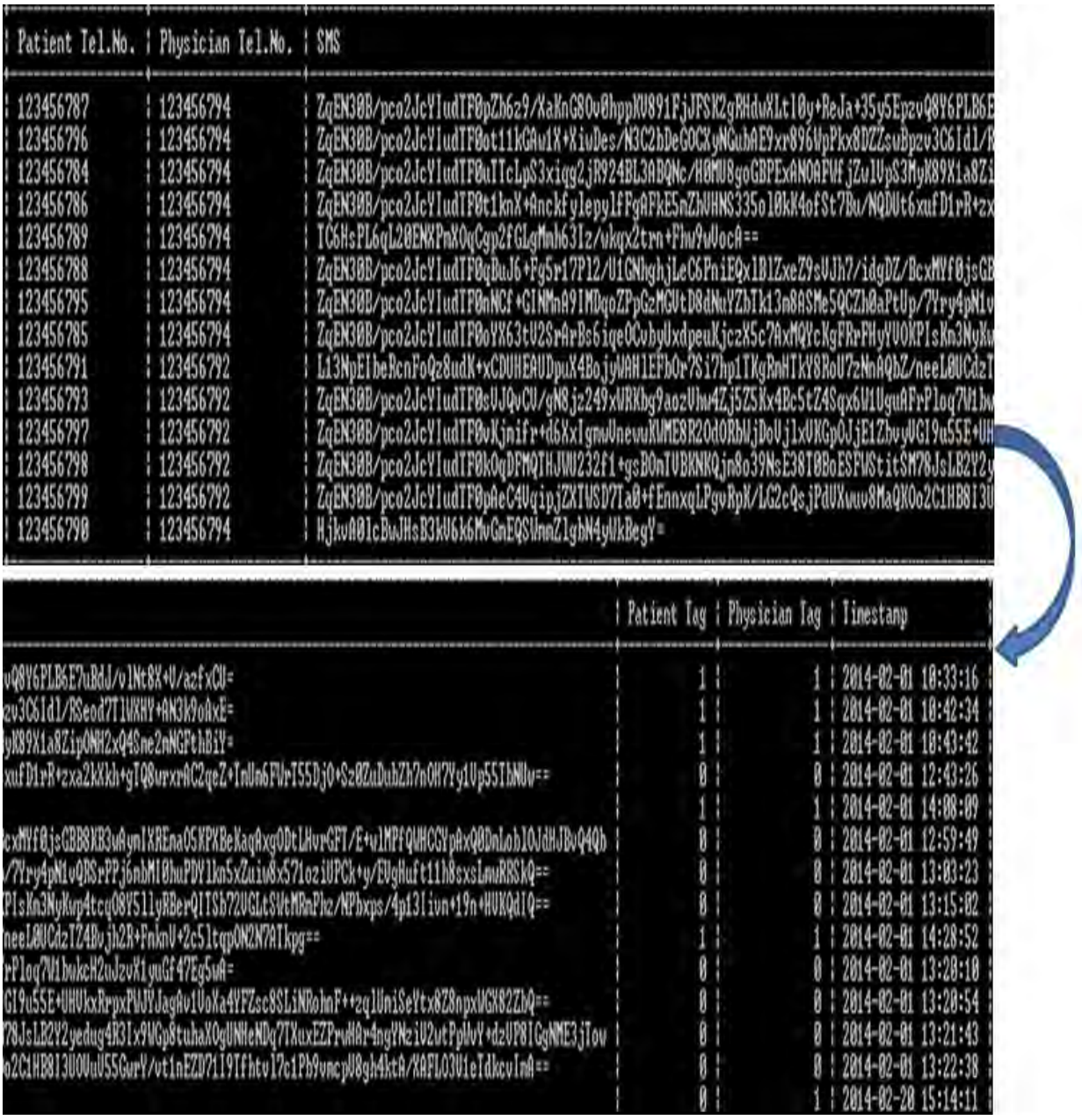
Appendix D: Part of the implementation of message decryption and display for the patient

```
public class showPmessage implements Runnable {
        public void start() {
            Thread t = new Thread(this);
            t.start();
        }
        public void run() {
          StringBuffer sb = new StringBuffer();
  try {
  HttpConnection c1 = (HttpConnection)
Connector.open("http://localhost:8080/sendMessage/pmessage");
                                                  ……..
                                                  ……..

                  os1.writeUTF(uname.trim());
                  os1.writeUTF(duname.trim());
                  os1.flush();  os1.close();
                    // get message from server
 DataInputStream is1 =(DataInputStream)c1.openDataInputStream();
                  int ch;       sb = new StringBuffer();
                while ((ch = is1.read()) != -1)
                 {   sb.append((char)ch);  }

                sms[index] = sb.toString();
                    Encryptor obj = new Encryptor(k1);    byte[] base64 = null;
                   base64 = Base64.decode(sms[index].getBytes("UTF-8"));
                //decrypt each message and keep in the string array
                 decryptedsms[index] = obj.decryptString(base64);
                                  is1.close();    c1.close();
                 } catch (Exception e) {
                    screen.append(new StringItem(" ",e.getMessage()));
                   }
              }
        }
}//End of showPmessage class
```

Appendix E: Part of the implementation of message decryption and display for the physician

```
public class message implements Runnable {
        public message() { }
        public void start(){
           Thread t = new Thread(this);
            t.start();
         }
        public void run() {
          StringBuffer sb = new StringBuffer();
            try {
 HttpConnection c1 = (HttpConnection)
Connector.open("http://localhost:8080/sendMessage/message");
c1.setRequestProperty("User-Agent","Profile/MIDP-1.0, Configuration/CLDC-1.0");
                c1.setRequestProperty("Content-Language","en-US");
                c1.setRequestMethod(HttpConnection.POST);
    DataOutputStream os1 =    (DataOutputStream)c1.openDataOutputStream();
                String r = String.valueOf(commonindex);
                os1.writeUTF(r.trim());
                os1.writeUTF(uname.trim()); //phone number of physician
                int need=1;       String n = String.valueOf(need);    //task id
                DataInputStream is1 =(DataInputStream)c1.openDataInputStream();
                int ch;   sb = new StringBuffer();
                while ((ch = is1.read()) != -1){
                      sb.append((char)ch);
                 }
            sms[commonindex] = sb.toString();
            Encryptor obj = new    Encryptor(k1); byte[] base64 = null;
            base64 = Base64.decode(sms[commonindex].getBytes("UTF-8"));
            decryptedsms[commonindex] = obj.decryptString(base64);
                                                    .....
                                                    .....
     } catch (Exception e) {  screen.append(new StringItem(" ",e.getMessage())); }
   }

}//End of message class
```

Appendix F: Message sending and storing process for the use of both patients and physicians



Patient Tel.No.	Physician Tel.No.	SMS
123456787	123456794	ZqEN30B/pco2JcYludTF0pZh6z9/XaKnG8Ov0hppKU891FjJFSK2gRHduXLtl0y+ReJa+35y5EpzvQ8Y6PLB6E
123456796	123456794	ZqEN30B/pco2JcYludTF0ot11kGAu1X+XiuDes/N3C2bDeGOCXyNGubAE9xr896WpPkx8DZZsuBpzv3C6Idl/
123456784	123456794	ZqEN30B/pco2JcYludTF0uITcLpS3xiqg2jR924BL3ABQNc/H8MU8goGBPExANOAFWfjZulVpS3MyK89X1a8Zi
123456786	123456794	ZqEN30B/pco2JcYludTF0t1knX+AnckfylepylfFgAFkE5nZhUHNS335o10kK4ofSt7Bu/NQDUt6xufD1rR+zx
123456789	123456794	IC6HsPL6qL20ENKPnX0qCgp2fGLgMmh63Iz/ukqx2trn+Phu9uUocA==
123456788	123456794	ZqEN30B/pco2JcYludTF0qBuJ6+Fg5r17P12/U1GNhghjLeC6PniEQx1BlZxeZ9sUJh7/idgDZ/BcxMYf0jsGB
123456795	123456794	ZqEN30B/pco2JcYludTF0nNCf+GINMnA9IMDqoZPpGzMGUtD8dNuYZbIk13n8ASMe5QCZh0aPtUp/7Yry4pN1v
123456785	123456794	ZqEN30B/pco2JcYludTF0oYX63tU2Sr4rBs6iqeOCubyUxdpeuKjcz85c7AxMQYcKgFRxFHyYUOKPIsKm3NyKw
123456791	123456792	L13NpEIbeRcnFoQz8udK+xCDUHERUDpuX4BojyUHH1EFbOr7Si7hp1TKgRnHTkY8RoU7zNnAQbZ/neeL8UCdzI
123456793	123456792	ZqEN30B/pco2JcYludTF0sUJQvCU/gN8jz249xURKbq9aozUhu4Zj5Z5Kx4Bc5tZ4Sqx6U1UquAFrPlaq7Wlbw
123456797	123456792	ZqEN30B/pco2JcYludTF0vKjnifr+d6XxIgnuUnevuKUME8R2OdORbUjDoUjlxUKGpOJjE1ZbvyUGI9u55E+U
123456798	123456792	ZqEN30B/pco2JcYludTF0kOqDFMQTHJUWJ232f1+gsBOnTVBKNKQjn8o39NsE38T0BoESFWStitSM78JsLB2Y2y
123456799	123456792	ZqEN30B/pco2JcYludTF0pheC4UqipjZXTWSD7Ta8+fEnnxqLPgvRpK/LG2cQsjPdUXuuv8MaQKOo2C1HB8I3U
123456790	123456794	HjkvA0IcBwJHsB3kU6k6MvGnEQSUnnZlghN4yUkBegY=

	Patient Tag	Physician Tag	Timestamp
vQ8Y6PLB6E7uBdJ/vlNt8X+U/azfxCU=	1	1	2014-02-01 10:33:16
zv3C6Idl/RSeod7TlUXHY+AN3k9oNxE=	1	1	2014-02-01 10:42:34
yK89X1a8ZipONH2xQ4Sne2nNGFthBiY=	1	1	2014-02-01 10:43:42
xufD1rR+zxa2kXkh+gIQ8urxrAC2qeZ+InUn6FUrT55DjO+Sz0ZuDubZh7nOH7Yg1Vp55IbNUw==	0	0	2014-02-01 12:43:26
	1	1	2014-02-01 14:08:09
cxMYf0jsGBB8KB3uAymIXREna05KPKBeKaqAxgODtLHvrGFT/E+ulMPfQUHCGYpAx0Q0DnLobl0JdHJDvQ4Qb	0	0	2014-02-01 12:59:49
/7Yry4pN1vQBSrPPj6nbMI0huPDYlknSxZuiu8x571oziUPCk+y/IUgHuft1ih8sxsLmuRBSkQ==	0	0	2014-02-01 13:03:23
PIsKn3NyKup4tcqO8Y5llyKBerQITSb72UGLtSVtMRnPhz/NPbxps/4pI3Iivn+19n+HUKQdIQ==	0	0	2014-02-01 13:15:02
neeL8UCdzTZ4BujhZR+PnknV+2c5ltqpON2N7ATkpg==	1	1	2014-02-01 14:28:52
rPlaq7W1bnkcH2uJzvK1yuGf47Eg5uA=	0	0	2014-02-01 13:28:10
G19u55E+UHUkxRrpxPUUYJagAv1UoKa4YFZsc8SLiNRoJnF++zq1UniSeYtx8Z8npxdUGK82ZbQ==	0	0	2014-02-01 13:28:54
78JsLR2V2yedug4R3Ix9UGp8tuhaXOgUNHeNDq7TXuxEZPruH0r4ngYNziU2utPpUuY+dzUP8IGgNME3jIou	0	0	2014-02-01 13:21:43
b2C1HB8I3U0UuU55GurY/vtinEZD7i19TfhtvI7c1Pb9vncpU8gh4ktA/XAFLO3UieIdkcvInA==	0	0	2014-02-01 13:22:38
	0	1	2014-02-20 15:14:11

Appendix G: Patient profile including the codes for patients' privacy protection purpose

EncodeName	Tel.No.	F.Name	L.Name	DOB	Sex	Country	City	Kebele	HouseNo.	EncodeHouseNo	Date
IUB181	123456789	Almaz	Tefera	1980-10-04	F	Ethiopia	Awasa	03	125	XSA137	2013-12-25
SUS171	123456791	Beyene	Kebede	1990-11-05	M	Ethiopia	Awasa	03	125	XSA137	2013-12-25
HOL126	123456790	Yeshe	Menberu	1975-03-02	F	Ethiopia	Addis Ababa. Sub City: Arada	01	240	PMC152	2013-12-25
XOS133	123456793	Abera	Mamo	1992-09-07	M	Ethiopia	Addis Ababa. Sub City: Arada	01	240	PMC152	2013-12-25
no	123456784	Selomon	Beya	1974-12-05	M	Ethiopia	Addis Ababa. Sub City: Bole	04	305	no	2013-12-28
no	123456785	Hlina	Kuma	1978-05-03	F	Ethiopia	Bahirdar	02	233	no	2013-12-28
GUO192	123456786	Mesfin	Mekurea	1971-09-04	M	Ethiopia	Bahirdar	05	259	DXA195	2013-12-28
UMT132	123456787	Daniel	Mekurea	1975-09-04	M	Ethiopia	Bahirdar	05	259	DXA195	2013-12-28
GRJ146	123456788	Dawet	Henok	1969-09-03	M	Ethiopia	Addis Ababa. Sub City: Bole	04	305	FDM162	2013-12-28
FQI170	123456795	Feben	Damte	1976-07-01	F	Ethiopia	Awasa	03	128	FQI170	2013-12-28
LEI175	123456796	Hana	Getaneh	1968-09-03	F	Ethiopia	Addis Ababa. Sub City: Arada	01	245	LEI175	2013-12-28
XHZ102	123456797	Dereje	Temesgen	1972-09-04	M	Ethiopia	Bahirdar	02	230	XHZ102	2013-12-28
CRS176	123456798	Mulgeta	Tefera	1974-08-03	M	Ethiopia	Bahirdar	05	259	DXA195	2013-12-28
MTK171	123456799	Zelalem	Feleke	1971-04-08	M	Ethiopia	Awasa	05	126	MTK171	2013-12-28

Appendix H: Patients set their privacy protection policy and stored in the database

User	Patient Tel.No.	Disease	CodeName	CodeHouseNo	DOB	Sex	Country	City	Kebele	Grant Date	Expire Date
Physician	123456790	HIV	0	0	1	1	1	1	0	2014-01-16	2014-02-04
Physician	123456790	PCP	0	0	1	1	1	1	1	2014-01-16	2014-02-03
Family	123456790	HIV	0	0	1	1	1	1	1	2013-12-28	2014-03-01
Family	123456790	PCP	0	0	1	1	1	1	1	2013-12-28	2014-03-01
Payer Organization	123456790	HIV	0	0	1	1	1	1	1	2013-12-28	2014-03-01
Payer Organization	123456790	PCP	0	0	1	1	1	1	1	2013-12-28	2014-03-01
Court of Law	123456790	HIV	0	0	1	1	1	1	1	2013-12-28	2014-03-01
Court of Law	123456790	PCP	0	0	1	1	1	1	1	2013-12-28	2014-03-01
Medical Research	123456790	HIV	1	1	1	1	1	1	1	2013-12-28	2014-03-01
Medical Research	123456790	PCP	1	1	1	1	1	1	0	2014-01-16	2014-02-04
Hospital Accreditation	123456790	HIV	1	1	1	1	1	1	0	2014-01-16	2014-05-04
Hospital Accreditation	123456790	PCP	1	1	1	1	1	1	1	2013-12-28	2014-03-01
Public Health Management	123456790	PCP	1	1	1	1	1	1	0	2014-01-16	2014-05-07
Public Health Management	123456790	HIV	0	0	1	1	1	1	1	2013-12-28	2014-03-01
Physician	123456789	HIV	0	0	1	1	1	1	1	2013-12-28	2014-05-03
Family	123456789	HIV	0	0	1	1	1	1	1	2013-12-28	2014-05-03
Payer Organization	123456789	HIV	0	0	1	1	1	1	1	2013-12-28	2014-05-03
Court of Law	123456789	HIV	0	0	1	1	1	1	1	2013-12-28	2014-05-03
Medical Research	123456789	HIV	1	1	1	1	1	1	0	2014-02-01	2015-01-02
Hospital Accreditation	123456789	HIV	1	1	1	1	1	1	1	2013-12-28	2014-05-03
Public Health Management	123456789	HIV	0	0	1	1	1	1	1	2013-12-28	2014-05-03
Public Policy Makers	123456789	HIV	1	1	1	1	1	1	1	2013-12-28	2013-05-03
Public Policy Makers	123456790	HIV	1	1	1	1	1	1	1	2013-12-28	2014-03-04
Public Policy Makers	123456790	PCP	1	1	1	1	1	1	1	2013-12-28	2014-03-04
Physician	123456793	HIV	1	1	1	1	1	1	0	2014-01-28	2014-03-04
Physician	123456793	PCP	0	0	1	1	1	1	1	2013-12-28	2014-03-04
Family	123456793	HIV	0	0	1	1	1	1	1	2013-12-28	2014-03-04
Family	123456793	PCP	0	0	1	1	1	1	1	2013-12-28	2014-03-04
Payer Organization	123456793	HIV	1	1	1	1	1	1	0	2014-01-28	2014-04-03
Payer Organization	123456793	PCP	1	1	1	1	1	0	0	2014-01-28	2014-04-03
Court of Law	123456793	HIV	1	1	1	1	1	1	0	2014-01-23	2014-04-06
Court of Law	123456793	PCP	1	1	1	1	1	1	1	2014-01-22	2014-03-05
Medical Research	123456793	HIV	1	1	1	1	1	1	1	2014-02-01	2015-02-03
Medical Research	123456793	PCP	1	1	1	1	1	1	1	2013-12-28	2014-03-04
Hospital Accreditation	123456793	HIV	1	1	1	1	1	1	1	2013-12-28	2014-03-04
Hospital Accreditation	123456793	PCP	1	1	1	1	1	1	1	2013-12-28	2014-03-04
Public Health Management	123456793	HIV	1	1	1	1	1	1	1	2013-12-28	2014-03-04
Public Health Management	123456793	PCP	1	1	1	1	1	1	1	2013-12-28	2014-03-04
Public Policy Makers	123456793	HIV	1	1	1	1	1	1	0	2014-01-23	2014-01-22
Public Policy Makers	123456793	PCP	1	1	1	1	1	1	1	2013-12-28	2014-03-04
Physician	123456791	HIV	0	0	1	1	1	1	1	2013-12-28	2014-02-03
Family	123456791	HIV	0	0	1	1	1	1	1	2013-12-28	2014-02-03
Payer Organization	123456791	HIV	0	0	1	1	1	1	1	2013-12-28	2014-02-03
Court of Law	123456791	HIV	0	0	1	1	1	1	1	2013-12-28	2014-02-03
Medical Research	123456791	HIV	1	1	1	1	1	1	1	2014-02-01	2014-03-04
Hospital Accreditation	123456791	HIV	1	1	1	1	1	1	1	2013-12-28	2014-02-03
Public Health Management	123456791	HIV	1	1	1	1	1	1	1	2013-12-28	2014-02-03
Public Policy Makers	123456791	HIV	1	1	1	1	1	1	1	2013-12-28	2014-02-03
Physician	123456787	HIV	0	0	1	1	1	1	1	2013-12-28	2014-02-04
Physician	123456787	MAC	0	0	1	1	1	1	1	2013-12-28	2014-02-04
Family	123456787	HIV	0	0	1	1	1	1	1	2013-12-28	2014-02-04
Family	123456787	MAC	0	0	1	1	1	1	1	2013-12-28	2014-02-04

Appendix I: Medical history of patients is available in the database

Patient Tel.No.	Disease	Treatment	Date
123456790	HIV	ART	2013-09-10
123456790	PCP	anti-pcp	2013-12-09
123456789	HIV	ART	2013-07-08
123456793	HIV	ART	2013-06-05
123456793	PCP	anti-pcp	2013-12-03
123456791	HIV	ART	2013-11-09
123456787	HIV	ART	2013-12-28
123456787	MAC	Anti-MAC	2013-12-28
123456788	HIV	ART	2013-12-28
123456788	MAC	Anti-MAC	2013-12-28
123456795	HIV	ART	2013-12-28
123456795	MAC	Anti-MAC	2013-12-28
123456796	HIV	ART	2013-12-28
123456796	MAC	Anti-MAC	2013-12-28
123456797	HIV	ART	2013-12-28
123456797	MAC	Anti-MAC	2013-12-28
123456798	HIV	ART	2013-12-28
123456798	MAC	Anti-MAC	2013-12-28
123456799	HIV	ART	2013-12-28
123456799	MAC	Anti-MAC	2013-12-28
123456784	HIV	ART	2013-12-28
123456785	HIV	ART	2013-12-28
123456786	HIV	ART	2013-12-28

Appendix J: Implementation of the Moxi z emulator

```
public class Sensor extends Thread {
   static final int MAXQUEUE = 3;
   private Vector messages = new Vector();
   //@Override
   public void run() {
      try {
         while ( true ){
            putMessage();
           sleep(1000 );
          }
      }
      catch( InterruptedException e )
      {System.out.println(e); }
   } //End of run

   private synchronized void putMessage() throws InterruptedException {
      while ( messages.size() == MAXQUEUE )
      wait();
      int k = new Random().nextInt(1600); // CD4 count can reach 1600 cells/mm3
      messages.addElement( new Integer(k));
      notifyAll();
   }//End of putMessage

   public synchronized int getMessage() throws InterruptedException {
      notify();
      while ( messages.size() == 0 )
        {  wait();  }
         Integer msg = (Integer)messages.firstElement();
         messages.removeElement(msg);
         int msg1 = msg.intValue();
      return msg1;
}//End of getMessage
} //End of Sensor class
```

Appendix K: Part of the implementation of CD4 count interpreter and message generator

```
public class Preprocessor extends Thread {
   SensorPMobile.Sensor producer;
   String name;
   SensorPMobile midlet;
   private Display display;
  public Preprocessor(String n,SensorPMobile.Sensor p,SensorPMobile midlet) {
   producer = p;
   name = n;
   this.midlet = midlet;
   display = Display.getDisplay(midlet);
   }
   public void start() {
           Thread t = new Thread(this);
           t.start();
    }
  // @Override
   public void run() {
      try {

          while ( true ) {
    int message = producer.getMessage(); // listen the CD4 count via the sensor instance
           String mess1 = "";
           String url = "http://localhost:8080/sendMessage/getConnection";
           StringBuffer sb = new StringBuffer();
          try {

            if(message<=350){
              HttpConnection c = (HttpConnection) Connector.open(url);
    c.setRequestProperty("User-Agent","Profile/MIDP-1.0, Configuration/CLDC-1.0");
```

```
c.setRequestProperty("Content-Language","en-US");
            c.setRequestMethod(HttpConnection.POST);
            DataOutputStream os = (DataOutputStream)c.openDataOutputStream();

  if(message<200){

              if(message<75){

                mess1 = "The CD4 count of "+nameofUser+" has reached
"+message+" cells/mm3, check for AIDS and Mycobacterium avium
complex[MAC]";
              }
              else if(message<100) {  //[75,100)

                 mess1 = "The CD4 count of "+nameofUser+" has reached
"+message+" cells/mm3, check for AIDS and for Toxoplasmosis and
Cryptococcosis";
              }
              else  {     //[100,200)
                 mess1 = "The CD4 count of "+nameofUser+" has reached
"+message+" cells/mm3, check for AIDS and for Pneumocystis pneumonia[PCP]";
              }
  }

           else if(message<300) {    // CD4 count is [200,300)
              mess1 = "The CD4 count of "+nameofUser+" has reached "+message+"
cells/mm3, Please initiate ART"+" and  also initiate ARV to prevent PCP";
           }

           else {    // CD4 count is [300,350)
              mess1 = "The CD4 count of "+nameofUser+" has reached "+message+"
cells/mm3, Please initiate ART";
           }

       String encdata; // save encrypted data on this
            Encryptor obj = new Encryptor(k1);
```

```
            byte[] encryptedbyte = obj.encryptString(mess1);
            byte[] enc64 = Base64.encode(encryptedbyte); //Base 64 encoder
            encdata = new String(enc64);
            os.writeUTF(encdata.trim());
            os.writeUTF(uname.trim());
            os.writeUTF(duname.trim());
            os.flush();
            os.close();
 DataInputStream is =(DataInputStream)c.openDataInputStream(); // Get response
from server
            int ch;     sb = new StringBuffer();
            while ((ch = is.read()) != -1)

           {   sb.append((char)ch);     }

            is.close();
            c.close();  } //End of if(message<350)

            } catch (Exception e) { System.out.println(e); }

           if(message<=350) {
              display.setCurrent(choose);
              showAlert(sb.toString()+" Check your page for Message later.");
              count = 120000;
            }
           else {
             count = 1000;
           }
           sleep(count); //sleeps before getting the next randum number
         }//End of while
      } catch( InterruptedException e )
          {System.out.println(e); }
  }//End of run
        private void showAlert(String sms) { Alert a = new Alert("");
        a.setString(sms);
       a.setTimeout(Alert.FOREVER);
       display.setCurrent(a); } //End of showAlert function
} //End of Preprocessor class
```

Appendix L: Part of the implementation of edited messages sent from physicians' mobile terminal

```
public class tag extends Thread {

        DoctorNotification midlet;
        private Display display;
        String name,mess;
        public tag(DoctorNotification midlet,String pname, String message) {
                                        .....
                                        .....
           String encdata = "";
           Encryptor obj = new Encryptor(k1);
           byte[] encryptedbyte = null;
        try {
           encryptedbyte = obj.encryptString(message);
           byte[] enc64 = Base64.encode(encryptedbyte);
           encdata = new String(enc64);

        } catch (CryptoException ex) { ex.printStackTrace(); }
             this.midlet = midlet;
            display = Display.getDisplay(midlet);
             name = pname;
             mess = encdata;
        }
        public void start() {
          display.setCurrent(mainscreen);
          mainscreen.deleteAll();
          mainscreen.append(new StringItem("   ", "Please Wait for a while...."));
          Thread t = new Thread(this);
           t.start();

       } //End of start function
```

```
public void run() {

            StringBuffer sb = new StringBuffer();
              try {

                    HttpConnection c = (HttpConnection)Connector.open
("http://localhost:8080/sendMessage/message1");
                                                    ……..
                                                    ……..

                    DataOutputStream os = (DataOutputStream)c.openDataOutputStream();
                                                    ……..
                                                    ……..

                    os.writeUTF(uname.trim()); // phone number of physician
                    os.writeUTF(name.trim()); // full name of patient
                    os.writeUTF(mess.trim()); // message of physician
                    os.flush();
                    os.close();
                    DataInputStream is =(DataInputStream)c.openDataInputStream();
                    int ch;
                    sb = new StringBuffer();
                    while ((ch = is.read()) != -1) {
                     sb.append((char)ch);
                     }
                   sleep(5000); // wait until the buffer have some message from the server
                    mainscreen.deleteAll();
                    mainscreen.append(new StringItem("",sb.toString()));
                    is.close();
                    c.close();
              } catch (Exception e) {
                    screen.append(new StringItem(" ",e.getMessage()));
                 }
          }
}//End of tag class
```

Appendix M: Part of the implementation of the privacy protection manager module

```
public class setaccesscontrol extends Thread {
          public void start() {
            Thread t = new Thread(this);
             t.start();
          }
          public void run() {
             StringBuffer sb = new StringBuffer();
  try {
           HttpConnection c1 = (HttpConnection) Connector.open
("http://localhost:8080/sendMessage/pmessageNo");
                                                  .........
                                                  .........
           os1.writeUTF(r.trim());                    os1.writeUTF(uname.trim());
           os1.writeUTF(duname.trim());          os1.writeUTF(usernames.trim());
           os1.writeUTF(encrypteduname.trim());
           os1.writeUTF(diseasename.trim());
           os1.writeUTF(String.valueOf(encodeindicator).trim());
           os1.writeUTF(String.valueOf(age).trim());
           os1.writeUTF(String.valueOf(sex).trim());
           os1.writeUTF(String.valueOf(country).trim());
           os1.writeUTF(String.valueOf(city).trim());
           os1.writeUTF(String.valueOf(kebele).trim());
           os1.writeUTF(expdate.trim());
           os1.flush(); os1.close();
            DataInputStream is1 =(DataInputStream)c1.openDataInputStream();
            int ch; int inserted = 0; sb = new StringBuffer();
             while ((ch = is1.read()) != -1)
              {    sb.append((char)ch); }
              inserted = Integer.parseInt(sb.toString());
              if(inserted!=0)
               {     attribute.append(new StringItem("","Saved Successfully"));      }
                                                  ..........
                                                  ..........
}//End of setaccesscontrol class
```

Appendix N: Part of the implementation for sharing medical information

```
public class medicalHistoryByDisease implements Runnable {
        public medicalHistoryByDisease() {      }
        public void start() {
             Thread t = new Thread(this);
              t.start();
         }
        public void run() {
          StringBuffer sb = new StringBuffer();
            try {
       HttpConnection c1 = (HttpConnection)
Connector.open("http://localhost:8080/sendMessage/message");
 c1.setRequestProperty("User-Agent","Profile/MIDP-1.0, Configuration/CLDC-1.0");
                  c1.setRequestProperty("Content-Language","en-US");
                  c1.setRequestMethod(HttpConnection.POST);
     DataOutputStream os1 = (DataOutputStream)c1.openDataOutputStream();
     String r = String.valueOf(0);
      os1.writeUTF(r.trim());
      os1.writeUTF(uname.trim()); //phone No. of Dr.
      int need=2; /*id for the task*/        String n = String.valueOf(need);
      os1.writeUTF(n.trim());   os1.writeUTF(vuser.trim());  os1.writeUTF(n.trim());
      os1.writeUTF(n.trim());   os1.writeUTF(n.trim());   os1.writeUTF(n.trim());
      os1.writeUTF(n.trim());    os1.writeUTF(n.trim());
      os1.writeUTF(vdisease1.trim());  os1.writeUTF(vpatient1.trim());
      os1.writeUTF(n.trim()); os1.flush(); os1.close();
          while ((ch = is1.read()) != -1) {
               if(ch!=42)
               {  sb.append((char)ch);    }
               Else {
                       details1.append(new StringItem(" ", sb.toString()));
                        sb = new StringBuffer();
                 }
           }
                                                ……..
                                                ……..
}//End of medicalHistoryByDisease class
```

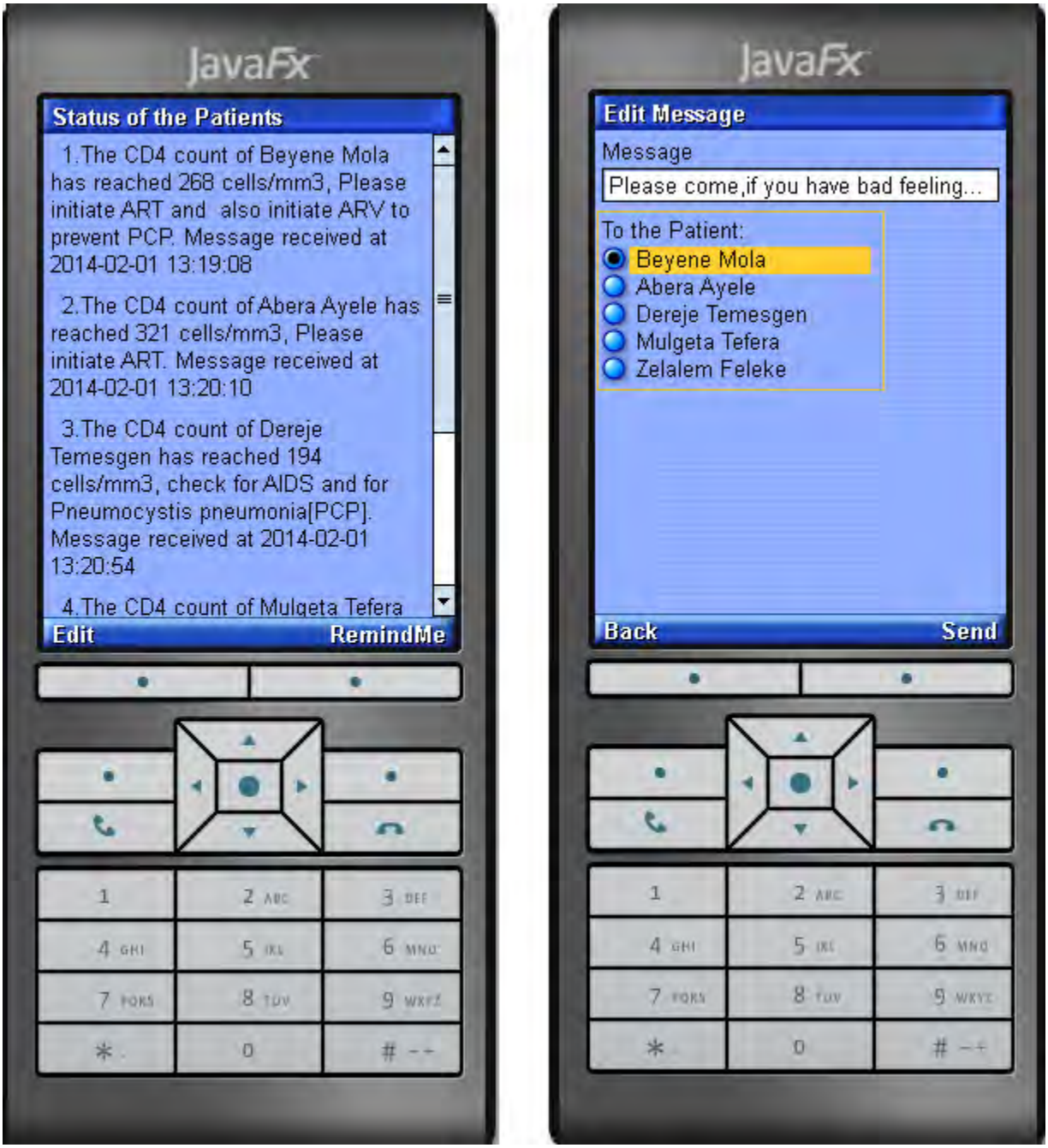
Appendix O: Part of implementation to get individual patients medical history

```
public class medicalHistoryByPatient implements Runnable {
        public medicalHistoryByPatient() {  }
        public void start(){
           Thread t = new Thread(this);
            t.start();
         }
        public void run() {
        StringBuffer sb = new StringBuffer();
        try {
            HttpConnection c1 = (HttpConnection)
Connector.open("http://localhost:8080/sendMessage/message");
                                                    .....
                                                    .....
            DataOutputStream os1 = (DataOutputStream)c1.openDataOutputStream();
            String r = String.valueOf(0);
            os1.writeUTF(r.trim());
            os1.writeUTF(uname.trim());
            int need=3;    String n = String.valueOf(need);
            os1.writeUTF(vpatient2.trim());
            os1.flush();     os1.close();
            DataInputStream is1 =(DataInputStream)c1.openDataInputStream();
            int ch;    sb = new StringBuffer(); details2.deleteAll();
            while ((ch = is1.read()) != -1) {
            if(ch!=42)
            { sb.append((char)ch);  }
            else {
              details2.append(new StringItem(" ", sb.toString()));
                sb = new StringBuffer();
             }
           }
           is1.close();       c1.close();
        } catch (Exception e) { screen.append(new StringItem(" ",e.getMessage()));
    }
 }
}//End of medicalHistoryByPatient class
```

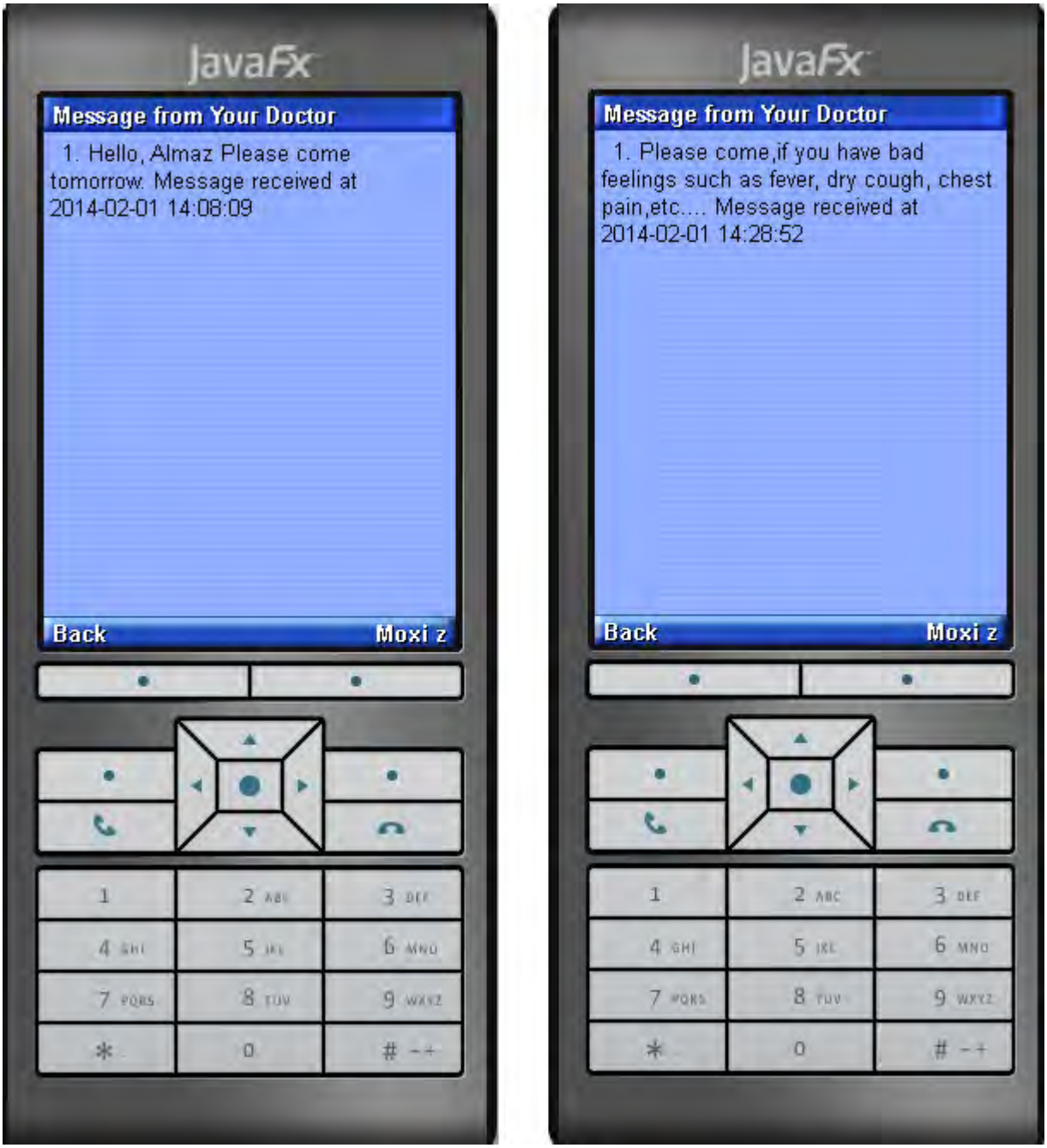
Appendix P: Part of the implementation of data export

```
public class ExportMyPatientsMedicalDetails implements Runnable {
        public ExportMyPatientsMedicalDetails()
        {  }
        public void start(){
           Thread t = new Thread(this);
           t.start();
        }
        public void run() {
          StringBuffer sb = new StringBuffer();
            try {
 HttpConnection c1 = (HttpConnection)
Connector.open("http://localhost:8080/sendMessage/message");
 c1.setRequestProperty("User-Agent","Profile/MIDP-1.0, Configuration/CLDC-1.0");
                                          ……..
                                          ……..
     DataOutputStream os1 = (DataOutputStream)c1.openDataOutputStream();
                os1.writeUTF(r.trim());     os1.writeUTF(uname.trim());
                int need=11;
                String n = String.valueOf(need);
                os1.writeUTF(n.trim());     os1.writeUTF(nameoffile.trim());
                                          ……..
                                          ……..
     DataInputStream is1 =(DataInputStream)c1.openDataInputStream();
                int ch;  sb = new StringBuffer();
                int i=0;
                while ((ch = is1.read()) != -1)
                { sb.append((char)ch); }//end of while

                if(sb.toString().equalsIgnoreCase("Yes"))
                { showAlert("Saved Successfully"); }
                else
                {   showAlert("Error Message: "+sb.toString());  }
                                          ……..
                                          ……..
}//End of ExportMyPatientsMedicalDetails class
```

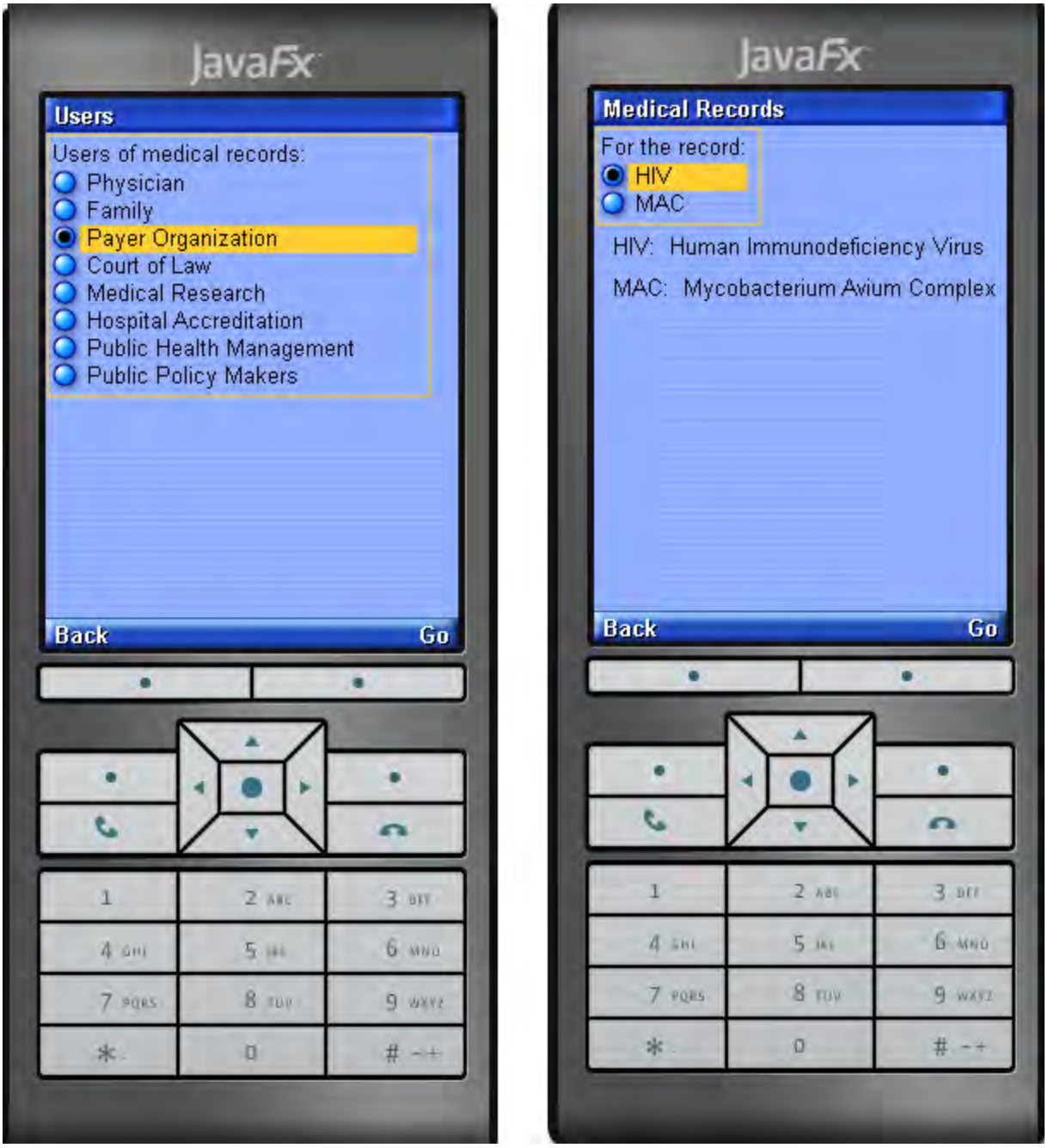
Appendix Q: Messages arrived at Dr. Abebe's smartphone



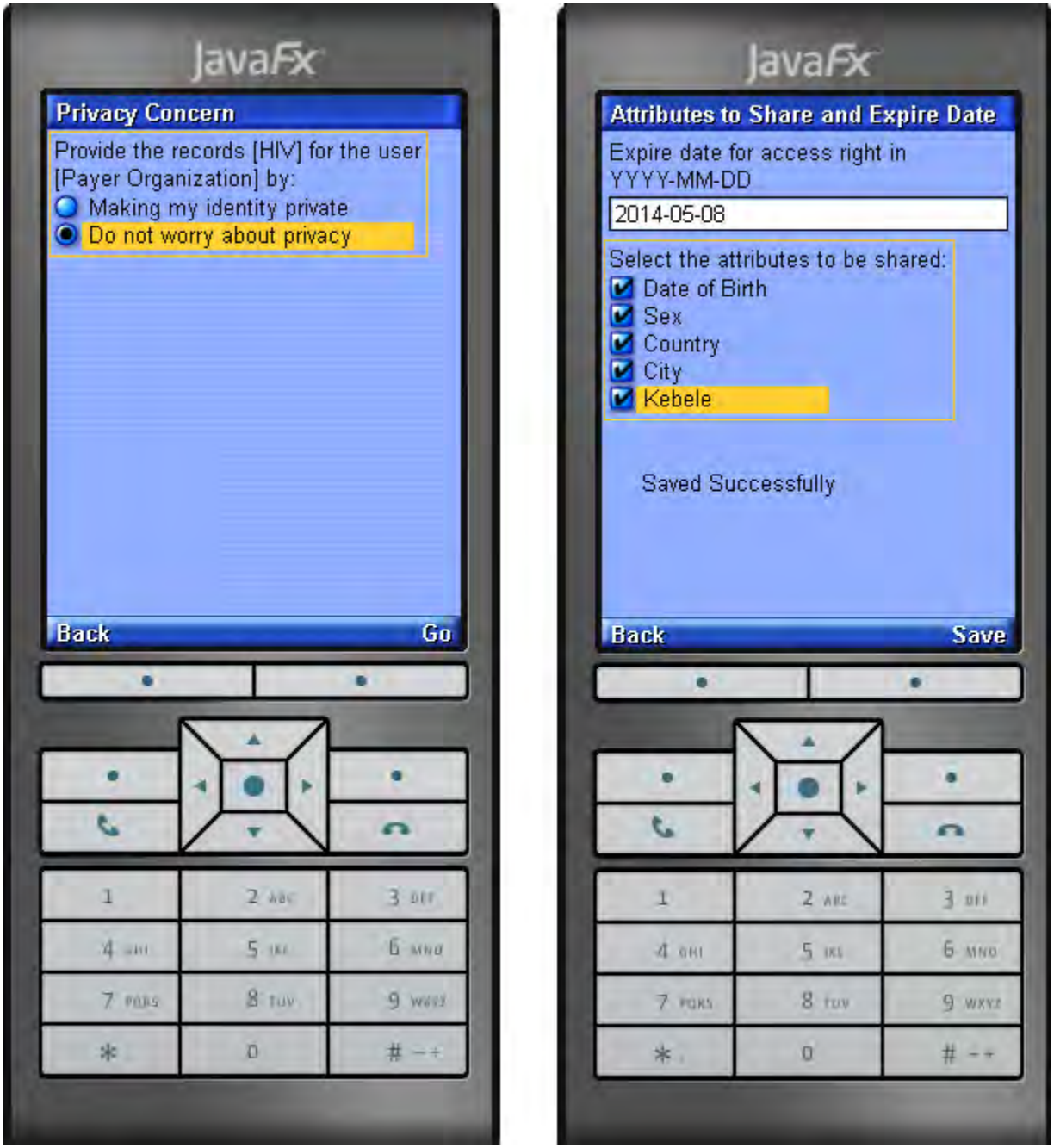
Appendix R: Messages arrived at Almaz's and Beyene's smartphones



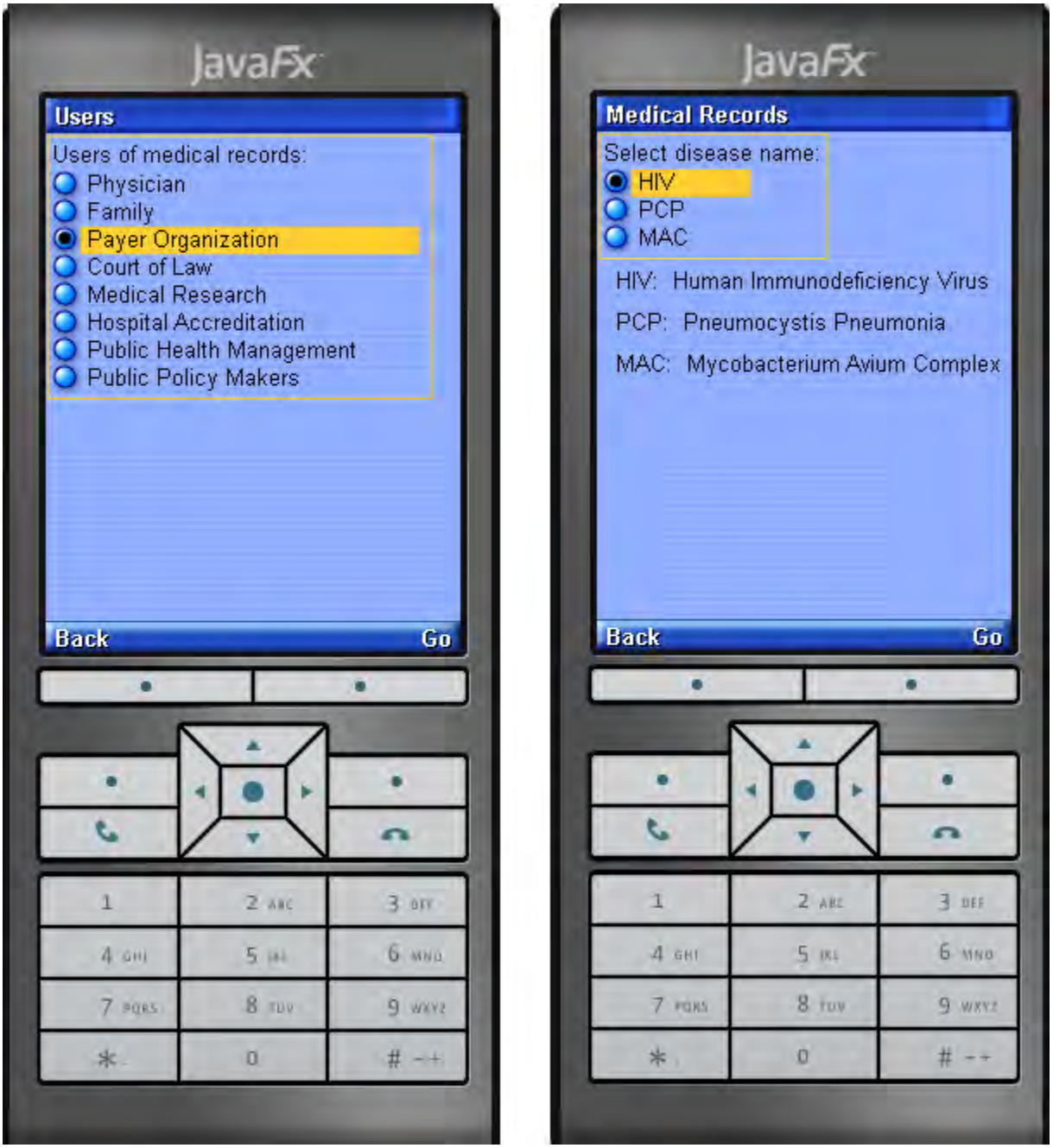
Appendix S: User and resource selection to set policy for payer organization



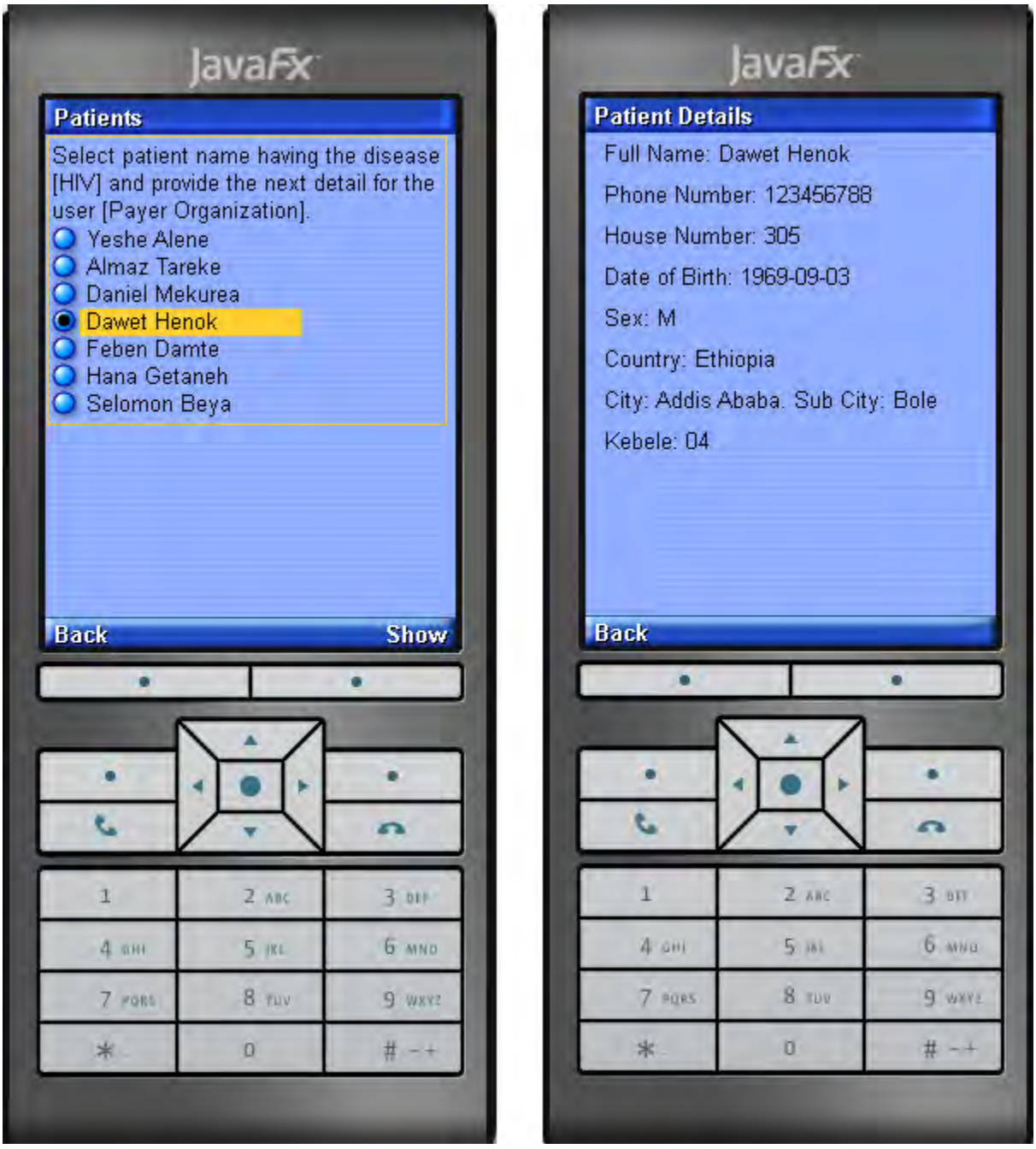
Appendix T: Setting policy on HIV for the use of payer organization



Appendix U: The payer organization's need of resource



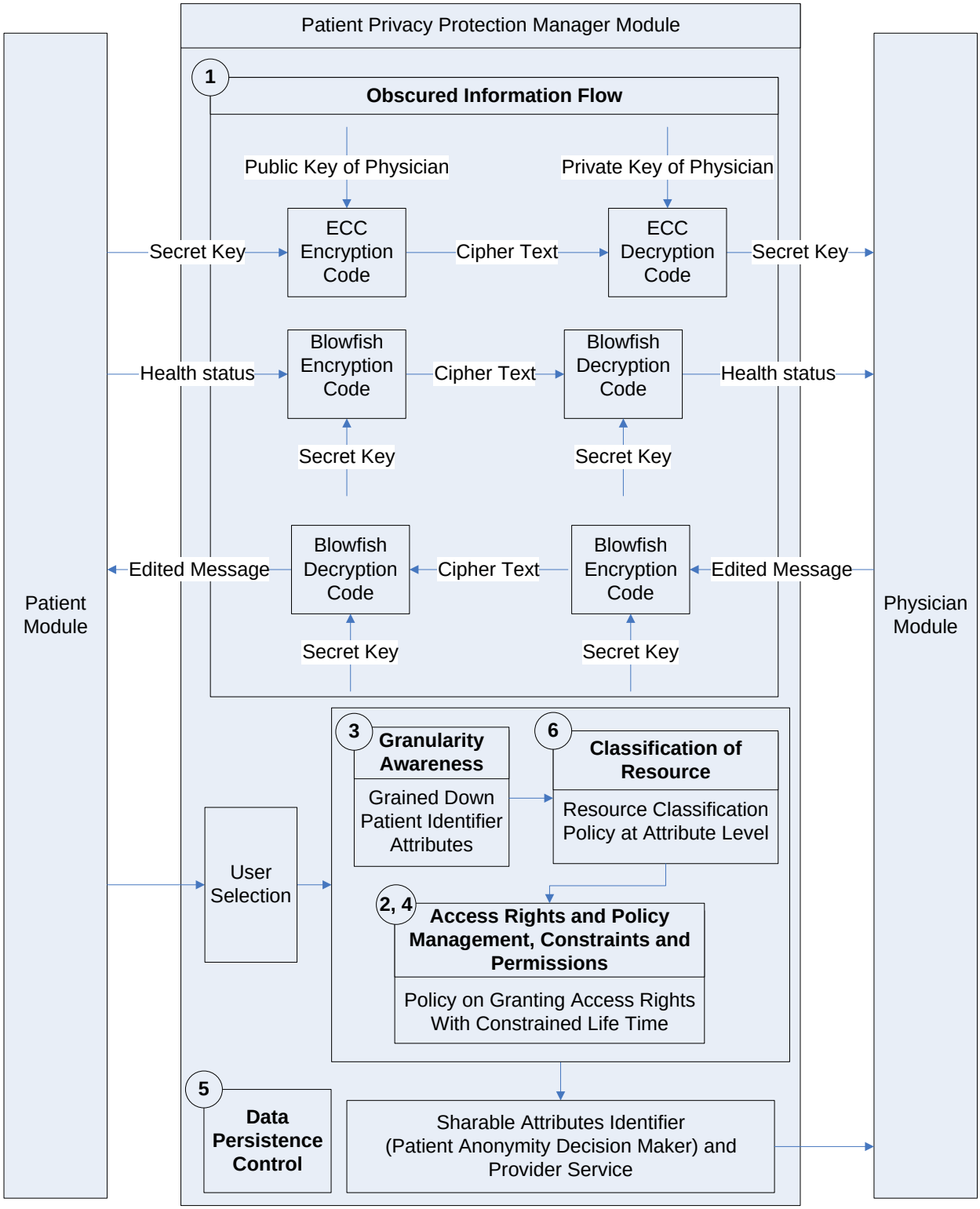
Appendix V: Detail information of Dawet Henok that are allowed for payer organization



Appendix W: Detail medical information of Dr. Tenna's patients

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Code for Name	Code for House Number	First Name	Last Name	Phone Number	Date of Birth	Sex	Country	City	Kebele	House Number	Type of Disease	Treatment	Date
2	HOL126	PMC152	Yeshe	Menberu	123456790	3/2/1975	F	Ethiopia	Addis Ababa. Sub City: Arada	1	240	HIV	ART	9/10/2013
3	HOL126	PMC152	Yeshe	Menberu	123456790	3/2/1975	F	Ethiopia	Addis Ababa. Sub City: Arada	1	240	PCP	anti-pcp	12/9/2013
4	IUB181	KSA137	Almaz	Tefera	123456789	10/4/1980	F	Ethiopia	Awasa	3	125	HIV	ART	7/8/2013
5	UMT132	DXA195	Daniel	Mekurea	123456787	9/4/1975	M	Ethiopia	Bahirdar	5	259	HIV	ART	12/28/2013
6	UMT132	DXA195	Daniel	Mekurea	123456787	9/4/1975	M	Ethiopia	Bahirdar	5	259	MAC	Anti-MAC	12/28/2013
7	GRJ146	FDM162	Dawet	Henok	123456788	9/3/1969	M	Ethiopia	Addis Ababa. Sub City: Bole	4	305	HIV	ART	12/28/2013
8	GRJ146	FDM162	Dawet	Henok	123456788	9/3/1969	M	Ethiopia	Addis Ababa. Sub City: Bole	4	305	MAC	Anti-MAC	12/28/2013
9	FQT170	AOQ133	Feben	Damte	123456795	7/1/1976	F	Ethiopia	Awasa	3	128	HIV	ART	12/28/2013
10	FQT170	AOQ133	Feben	Damte	123456795	7/1/1976	F	Ethiopia	Awasa	3	128	MAC	Anti-MAC	12/28/2013
11	LEI175	ECT151	Hana	Getaneh	123456796	9/3/1968	F	Ethiopia	Addis Ababa. Sub City: Arada	1	245	HIV	ART	12/28/2013
12	LEI175	ECT151	Hana	Getaneh	123456796	9/3/1968	F	Ethiopia	Addis Ababa. Sub City: Arada	1	245	MAC	Anti-MAC	12/28/2013
13	no	no	Selomon	Beya	123456784	12/5/1974	M	Ethiopia	Addis Ababa. Sub City: Bole	4	305	HIV	ART	12/28/2013
14	no	no	Hlina	Kuma	123456785	5/3/1978	F	Ethiopia	Bahirdar	2	233	HIV	ART	12/28/2013
15	GUO192	DXA195	Mesfin	Mekurea	123456786	9/4/1971	M	Ethiopia	Bahirdar	5	259	HIV	ART	12/28/2013

Appendix X: The more detail architecture of the patient privacy protection manager module



Declaration

I declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been acknowledged.

Declared by:

Name: _____

Signature: _____

Date: _____

Confirmed by Advisor:

Name: _____

Signature: _____

Date: _____

Place and date of submission: Addis Ababa, April, 2014