
International Journal of Advanced Multidisciplinary Research (IJAMR)

ISSN: 2393-8870

www.ijarm.com

Research Article

Secured privacy preserving opportunistic frame work for Mobile Healthcare Emergency

G.Yogeshwaran* and C.Gunaseelan

Department of Computer Applications, Thirumalai Engineering College, Kilambi, Kancipuram, Tamilnadu, India

*Corresponding Author

Abstract

Keywords

mobile Healthcare,
SPOC,
PPSPC,
PHI data,
m-Healthcare emergency.

With the pervasiveness of smart phones and the advance of wireless body sensor networks (BSNs), mobile Healthcare (m-Healthcare), which extends the operation of Healthcare provider into a pervasive environment for better health monitoring, has attracted considerable interest recently. However, the flourish of m-Healthcare still faces many challenges including information security and privacy preservation. In this paper, we propose a secure and privacy-preserving opportunistic computing framework, called SPOC, for m-Healthcare emergency. With SPOC, smart phone resources including computing power and energy can be opportunistically gathered to process the computing intensive personal health information (PHI) during m-Healthcare emergency with minimal privacy disclosure. In specific, to leverage the PHI privacy disclosure and the high reliability of PHI process and transmission in m-Healthcare emergency, we introduce an efficient user-centric privacy access control in SPOC framework, which is based on an attribute-based access control and a new privacy-preserving scalar product computation (PPSPC) technique, and allows a medical user to decide who can participate in the opportunistic computing to assist in processing his overwhelming PHI data. Detailed security analysis shows that the proposed SPOC framework can efficiently achieve user-centric privacy access control in m-Healthcare emergency. In addition, performance evaluations via extensive simulations demonstrate the SPOC's effectiveness in term of providing high reliable PHI process and transmission while minimizing the privacy disclosure during m-Healthcare emergency

Introduction

In our aging society, mobile Healthcare (m-Healthcare) system has been envisioned as an important application of pervasive computing to improve health care quality and save lives, where miniaturized wearable and implantable body sensor nodes and Smartphone's are utilized to provide remote healthcare monitoring to people who have chronic medical conditions such as diabetes and heart disease. Specifically, in an m-Healthcare system, medical users are no longer needed to be monitored within home or hospital environments. Instead, after being equipped with smart phone and wireless body sensor network (BSN) formed by body sensor nodes, medical users can walk outside and receive the high-quality healthcare monitoring from medical professionals anytime and anywhere. For example, as shown in Fig. 1, each mobile medical user's personal health information (PHI) such as heart beat, blood sugar level, blood pressure and temperature and others, can be first collected by BSN, and then aggregated by

smart phone via Bluetooth. Finally, they are further transmitted to the remote healthcare center via 3G networks. Based on these collected PHI data, medical professionals at healthcare center can continuously monitor medical users' health conditions and as well quickly react to users' life-threatening situations and save their lives by dispatching ambulance and medical personnel to an emergency location in a timely fashion.

Although m-Healthcare system can benefit medical user by providing high quality pervasive healthcare monitoring, the flourish of m-Healthcare system still hinges upon how we fully understand and manage the challenges facing in m-Healthcare system, especially during a medical emergency. To clearly illustrate the challenges in m Healthcare emergency, we consider the following scenario. In general, a medical user's PHI should be reported to the healthcare center every 5

minutes for normal remote monitoring. However, when he has an emergency medical condition, for example, heart attack, his BSN becomes busy reading a variety of medical measures, such as heart rate, blood pressure, and as a result, a large amount of PHI data will be generated in a very short period of time, and they further should be reported every 10 seconds for high intensive monitoring before ambulance and medical personnel's arrival. However, since smart phone is not only used for healthcare monitoring, but also for other applications, i.e., phoning with friends, the smart phone's energy could be insufficient when an emergency takes place. Although this kind of unexpected event may happen with very low probability, i.e., 0.005, for a medical emergency, when we take into 10, 000 emergency cases into consideration, the average event number will reach 50, which is not negligible and explicitly indicates the reliability of m-Healthcare system is still challenging in emergency.

Recently, opportunistic computing, as a new pervasive computing paradigm, has received much attention. Essentially, opportunistic computing is characterized by exploiting all available computing resources in an opportunistic environment to provide a platform for the distributed execution of a computing-intensive task. For example once the execution of a task exceeds the energy and computing power available on a single node, other opportunistically contacted nodes can contribute to the execution of the original task by running a subset of task, so that the original task can be reliably performed. Obviously, opportunistic computing paradigm can be applied in m-Healthcare emergency to resolve the challenging reliability issue in PHI process. However, PHI are personal information and very sensitive to medical users, once the raw PHI data are processed in opportunistic computing, the privacy of PHI would be disclosed. Therefore, how to balance the high reliability of PHI process while minimizing the PHI privacy disclosure during the opportunistic computing becomes a challenging issue in m-Healthcare emergency. In this paper, we propose a new secure and privacy preserving opportunistic computing framework, called SPOC, to address this challenge. With the proposed SPOC framework, each medical user in emergency can achieve the user-centric privacy access control to allow only those qualified helpers to participate in the opportunistic computing to balance the high-reliability of PHI process and minimizing PHI privacy disclosure in m-Healthcare emergency. Specifically, the main contributions of this paper are threefold.

First, we propose SPOC, a secure and privacy-preserving opportunistic computing framework for m-Healthcare emergency. With SPOC, the resources available on other opportunistically contacted medical users' smart phones can be gathered together to deal with the computing intensive PHI process in emergency situation. Since the PHI will be disclosed during the process in opportunistic computing, to

minimize the PHI privacy disclosure, SPOC introduces a user-centric two-phase privacy access control to only allow those medical users who have similar symptoms to participate in opportunistic computing.

Second, to achieve user-centric privacy access control in opportunistic computing, we present an efficient attribute based access control and a novel non homomorphism encryption based privacy-preserving scalar product computation (PPSPC) protocol, where the attributed-based access control can help a medical user in emergency to identify other medical users, and PPSPC protocol can further control only those medical users who have similar symptoms to participate in the opportunistic computing while without directly revealing users' symptoms. Note that, although PPSPC protocols have been well studied in privacy-preserving data mining, yet most of them are relying on time-consuming homomorphic encryption technique. To the best of our knowledge, our novel non-homomorphic encryption based PPSPC protocol is the most efficient one in terms of computational and communication overheads.

Third, to validate the effectiveness of the proposed SPOC framework in m-Healthcare emergency, we also develop a custom simulator built in Java. Extensive simulation results show that the proposed SPOC framework can help medical users to balance the high-reliability of PHI process and minimizing the PHI privacy disclosure in m-Healthcare emergency.

ORGANIZATION PROFILE

At Innovetech Pro Solutions, We go beyond providing software solutions. We work with our client's technologies and business changes that shape their competitive advantages.

Founded in 2000, Innovetech Pro Solutions (P) Ltd. is a software and service provider that helps organizations deploy, manage, and support their business-critical software more effectively. Utilizing a combination of proprietary software, services and specialized expertise, Innovetech Pro Solutions (P) Ltd. helps mid-to-large enterprises, software companies and IT service providers improve consistency, speed, and transparency with service delivery at lower costs. Innovetech Pro Solutions (P) Ltd. helps companies avoid many of the delays, costs and risks associated with the distribution and support of software on desktops, servers and remote devices. Our automated solutions include rapid, touch-free deployments, ongoing software upgrades, fixes and security patches, technology asset inventory and tracking, software license optimization, application self-healing and policy management. At Innovetech Pro Solutions, we go beyond providing software solutions. We work with our clients' technologies and business processes that shape their competitive advantages.

ABOUT THE PEOPLE

As a team we have the prowess to have a clear vision and realize it too. As a statistical evaluation, the team has more than 40,000 hours of expertise in providing real-time solutions in the fields of Embedded Systems, Control systems, Micro-Controllers, c Based Interfacing, Programmable Logic Controller, VLSI Design And Implementation, Networking With C ++, java, client Server Technologies in .NET, Java,(J2EE\J2ME\J2SE\EJB),VB & VC++, Oracle and operating system concepts with LINUX.

OUR VISION

“Dreaming a vision is possible and realizing it is our goal”.

OUR MISSION

We have achieved this by creating and perfecting processes that are in par with the global standards and we deliver high quality, high value services, reliable and cost effective IT products to clients around the world.

CLIENTS

- Aray InfoTech
- Inquirre consultancy (U.S.A)
- K square consultancy pvt Ltd (U.S.A)
- Opal solutions
- Texlab Solutions
- Vertex Business Machines
- JM InfoTech

SYSTEM ANALYSIS

EXISTING SYSTEM

In Existing System, According to the sensex over the age of 65 is expected to hit 70 million by 2030, having doubled since 2000. Health care expenditures projected to rise to 15.9% by 2013. The cost of health care for the nation’s aging population has become a national concern are important for understanding how the opportunistic computing paradigm work when resources available on different nodes can be opportunistically gathered together to provide richer functionality, they have not considered the potential security and privacy issues existing in the opportunistic computing paradigm.

Disadvantages

Less privacy and less potential security.
Not an effective health care monitoring.

Proposed System

In our proposed SPOC framework aims at the security and privacy issues, and develops a user-centric privacy access control of opportunistic computing in m-Healthcare emergency.

Advantages

1. Shift from a clinic-oriented, centralized healthcare system to a patient-oriented, distributed healthcare system.
2. Reduce healthcare expenses through more efficient use of clinical resources and earlier detection of medical conditions.
3. Performance, Reliability, Scalability, Quality of Service, Privacy, Security.
4. More prone to failures, caused by power exhaustion, software and hardware faults, natural disasters, malicious attacks, and human errors etc.
- 5.

DEVELOPMENT ENVIRONMENT

HARDWARE REQUIREMENTS:

Processor	:	Intel Dual Core.
Hard Disk	:	60 GB.
Floppy Drive	:	1.44 Mb.
Monitor	:	LCD Colour.
Mouse	:	Optical Mouse.
RAM	:	1 GB.

SOFTWARE REQUIREMENTS:

Operating system	:	Windows XP.
Coding Language	:	C#.Net
Database	:	SQL Server 2005

SOFTWARE ENVIRONMENT FEATURES OF .NET

Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There’s no language barrier with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that components created in different languages can easily interoperate.

“.NET” is also the collective name given to various software components built upon the .NET platform. These will be both products (Visual Studio.NET and Windows.NET Server, for instance) and services (like Passport, .NET My Services, and so on).

THE .NET FRAMEWORK

The .NET Framework has two main parts:
1. The Common Language Runtime (CLR).

2. A hierarchical set of class libraries.

The CLR is described as the “execution engine” of .NET. It provides the environment within which programs run. The most important features are

- Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on.
- Memory management, notably including garbage collection.
- Checking and enforcing security restrictions on the running code.
- Loading and executing programs, with version control and other such features.
- The following features of the .NET framework are also worth description:

Managed Code

The code that targets .NET, and which contains certain extra information - “metadata” - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the CLR to guarantee, for instance, safe execution and interoperability.

Managed Data

With Managed Code comes Managed Data. CLR provides memory allocation and deallocation facilities, and garbage collection. Some .NET languages use Managed Data by default, such as C#, Visual Basic.NET and JScript.NET, whereas others, namely C++, do not. Targeting CLR can, depending on the language you’re using, impose certain constraints on the features available. As with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications - data that doesn’t get garbage collected but instead is looked after by unmanaged code.

Common Type System

The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other, by describing types in a common way. CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception handling. As well as ensuring that types are only used in appropriate ways, the runtime also ensures that code doesn’t attempt to access memory that hasn’t been allocated to it.

Common Language Specification

The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.

THE CLASS LIBRARY

.NET provides a single-rooted hierarchy of classes, containing over 7000 types. The root of the namespace is called System; this contains basic types like Byte, Double, Boolean, and String, as well as Object. All objects derive from System. Object. As well as objects, there are value types. Value types can be allocated on the stack, which can provide useful flexibility. There are also efficient means of converting value types to object types if and when necessary.

The set of classes is pretty comprehensive, providing collections, file, screen, and network I/O, threading, and so on, as well as XML and database connectivity.

The class library is subdivided into a number of sets (or namespaces), each providing distinct areas of functionality, with dependencies between the namespaces kept to a minimum.

LANGUAGES SUPPORTED BY .NET

The multi-language capability of the .NET Framework and Visual Studio .NET enables developers to use their existing programming skills to build all types of applications and XML Web services. The .NET framework supports new versions of Microsoft’s old favorites Visual Basic and C++ (as VB.NET and Managed C++), but there are also a number of new additions to the family.

Visual Basic .NET has been updated to include many new and improved language features that make it a powerful object-oriented programming language. These features include inheritance, interfaces, and overloading, among others. Visual Basic also now supports structured exception handling, custom attributes and also supports multi-threading.

Visual Basic .NET is also CLS compliant, which means that any CLS-compliant language can use the classes, objects, and components you create in Visual Basic .NET.

Managed Extensions for C++ and attributed programming are just some of the enhancements made to the C++ language. Managed Extensions simplify the task of migrating existing C++ applications to the new .NET Framework.

C# is Microsoft’s new language. It’s a C-style language that is essentially “C++ for Rapid Application Development”. Unlike other languages, its specification is just the grammar of the language. It has no standard library of its own, and instead has been designed with the intention of using the .NET libraries as its own.

Microsoft Visual J# .NET provides the easiest transition for Java-language developers into the world of XML Web Services and dramatically improves the interoperability of Java-language programs with existing software written in a variety of other programming languages.

Active State has created Visual Perl and Visual Python, which enable .NET-aware applications to be built in either Perl or Python. Both products can be integrated into the Visual Studio .NET environment. Visual Perl includes support for Active State’s Perl Dev Kit.

Other languages for which .NET compilers are available include

- FORTRAN
- COBOL
- Eiffel

Fig1 .Net Framework

ASP.NET XML SERVICES	WEB	Windows Forms
Base Class Libraries		
Common Language Runtime		
Operating System		

C#.NET is also compliant with CLS (Common Language Specification) and supports structured exception handling. CLS is set of rules and constructs that are supported by the CLR (Common Language Runtime). CLR is the runtime environment provided by the .NET Framework; it manages the execution of the code and also makes the development process easier by providing services.

C#.NET is a CLS-compliant language. Any objects, classes, or components that created in C#.NET can be used in any other CLS-compliant language. In addition, we can use objects, classes, and components created in other CLS-compliant languages in C#.NET .The use of CLS ensures complete interoperability among applications, regardless of the languages used to create the application.

CONSTRUCTORS AND DESTRUCTORS

Constructors are used to initialize objects, whereas destructors are used to destroy them. In other words,

destructors are used to release the resources allocated to the object. In C#.NET the sub finalize procedure is available. The sub finalize procedure is used to complete the tasks that must be performed when an object is destroyed. The sub finalize procedure is called automatically when an object is destroyed. In addition, the sub finalize procedure can be called only from the class it belongs to or from derived classes.

GARBAGE COLLECTION

Garbage Collection is another new feature in C#.NET. The .NET Framework monitors allocated resources, such as objects and variables. In addition, the .NET Framework automatically releases memory for reuse by destroying objects that are no longer in use.

In C#.NET, the garbage collector checks for the objects that are not currently in use by applications. When the garbage collector comes across an object that is marked for garbage collection, it releases the memory occupied by the object.

OVERLOADING

Overloading is another feature in C#. Overloading enables us to define multiple procedures with the same name, where each procedure has a different set of arguments. Besides using overloading for procedures, we can use it for constructors and properties in a class.

OBJECTIVES OF .NET FRAMEWORK

1. To provide a consistent object-oriented programming environment whether object codes is stored and executed locally on Internet-distributed, or executed remotely.
2. To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code.
3. Eliminates the performance problems.

FEATURES OF SQL SERVER

The OLAP Services feature available in SQL Server version 7.0 is now called SQL Server 2000 Analysis Services. The term OLAP Services has been replaced with the term Analysis Services. Analysis Services also includes a new data mining component. The Repository component available in SQL Server version 7.0 is now called Microsoft SQL Server 2000 Meta Data Services. References to the component now use the term Meta Data Services. The term repository is used only in reference to the repository engine within Meta Data Services

SQL-SERVER database consist of six type of objects, They are,

1. TABLE
2. QUERY
3. FORM
4. REPORT
5. MACRO

TABLE:

A database is a collection of data about a specific topic.

VIEWS OF TABLE:

We can work with a table in two types,

1. Design View
2. Datasheet View

Design View

To build or modify the structure of a table we work in the table design view. We can specify what kind of data will be hold.

Datasheet View

To add, edit or analyses the data itself we work in tables datasheet view mode.

QUERY:

A query is a question that has to be asked the data. Access gathers data that answers the question from one or more table. The data that make up the answer is either dynaset (if you edit it) or a snapshot (it cannot be edited).Each time we run query, we get latest information in the dynaset. Access either displays the dynaset or snapshot for us to view or perform an action on it, such as deleting or updating.

AJAX:

ASP.NET Ajax marks Microsoft's foray into the ever-growing Ajax framework market. Simply put, this new environment for building Web applications puts Ajax at the front and center of the .NET Framework.

PROJECT DESIGN STAGES

LIST OF MODULES

1. Health monitoring in M-Healthcare
2. Body Sensor Network
3. Security Analysis
4. Performance Evolution
5. Report Generation

MODULES DESCRIPTION

HEALTH MONITORING IN M-HEALTHCARE

In this module, each mobile medical user’s personal health information (PHI) such as heart beat, blood sugar level, blood pressure and temperature and others, can be first collected by BSN, and then aggregated by smart phone via Bluetooth. Finally, they are further transmitted to the remote healthcare center via 3G networks. Based on these collected PHI data, medical professionals at healthcare center can continuously monitor medical users’ health conditions and as well quickly react to users’ life-threatening situations and

save their lives by dispatching ambulance and medical personnel to an emergency location in a timely fashion.

BODY SENSOR NETWORK

In this module, Body area network (BAN), wireless body area network (WBAN) or body sensor network (BSN) are terms used to describe the application of wearable computing devices. This will enable wireless communication between several miniaturized body sensor units (BSU) and a single body central unit (BCU) worn at the human body.

- Deploy wearable sensors on the bodies of patients in a residential setting
- Continuously monitor physiological signals (such as ECG, blood oxygen levels) and other health related information (such as physical activity)

SECURITY ANALYSIS

In this Module to develop a secure and privacy-preserving opportunistic computing framework to provide high reliability of PHI process and transmission while minimizing PHI privacy disclosure in m-Healthcare emergency. Specifically, we

- apply opportunistic computing in m-Healthcare emergency to achieve high-reliability of PHI process and transmission; and
- develop user-centric privacy access control to minimize the PHI privacy disclosure.

PERFORMANCE EVOLUTION

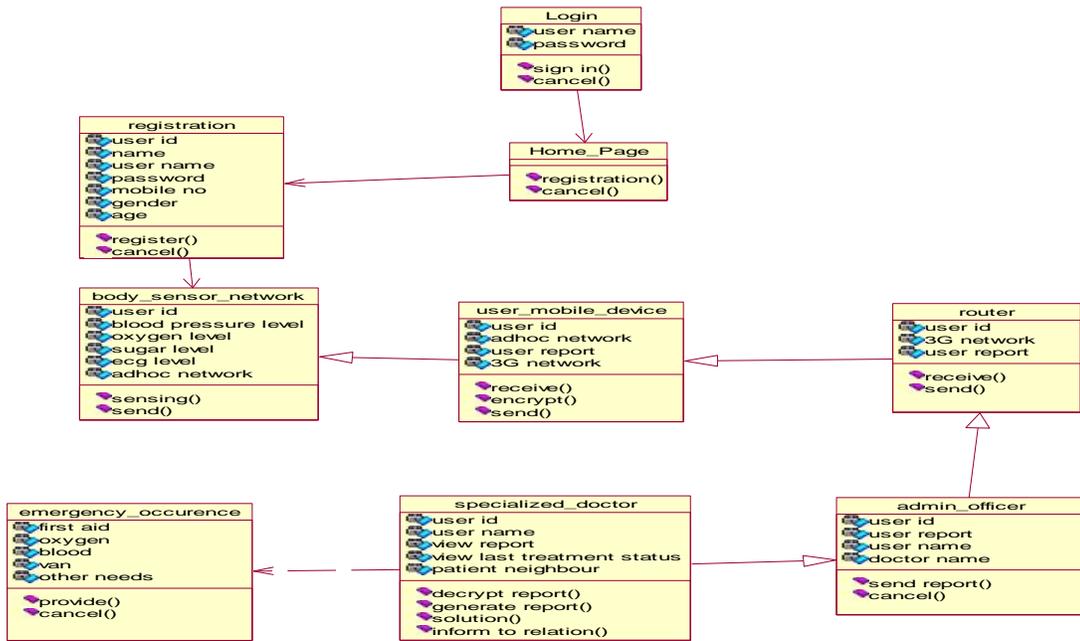
In this module, the performance metrics used in the evaluation are :

- 1) The average number of qualified helpers (NQH), which indicates how many qualified helpers can participate in the opportunistic computing within a given time period, and
- 2) The average resource consumption ratio (RCR), which is defined as the fraction of the resources consumed by the medical user in emergency to the total resources consumed in opportunistic computing for PHI process within a given time period.

REPORT GENERATION

In this module, Health care center generate crystal report from the database collection for future reference.

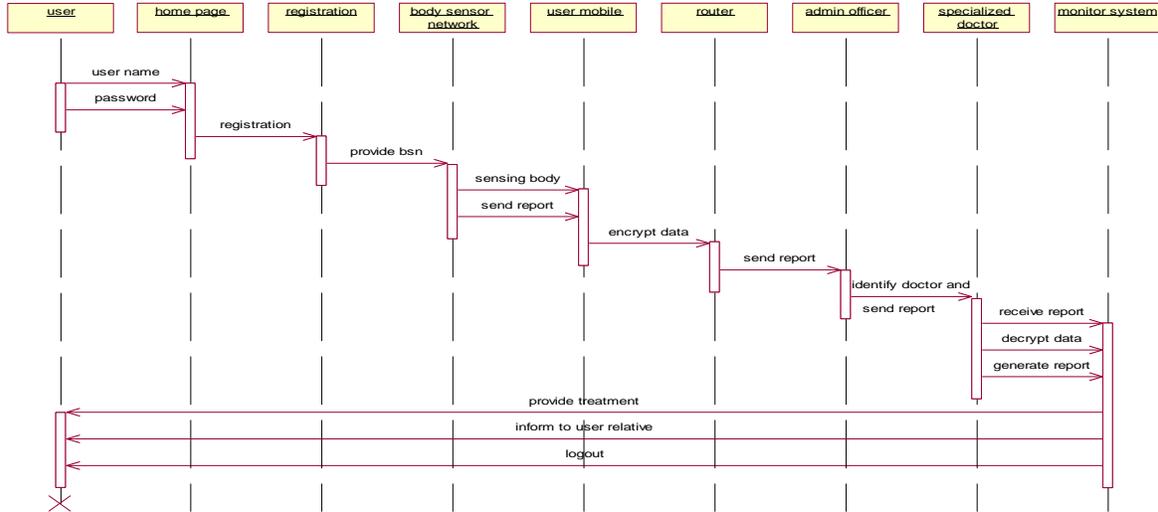
CLASS DIAGRAM



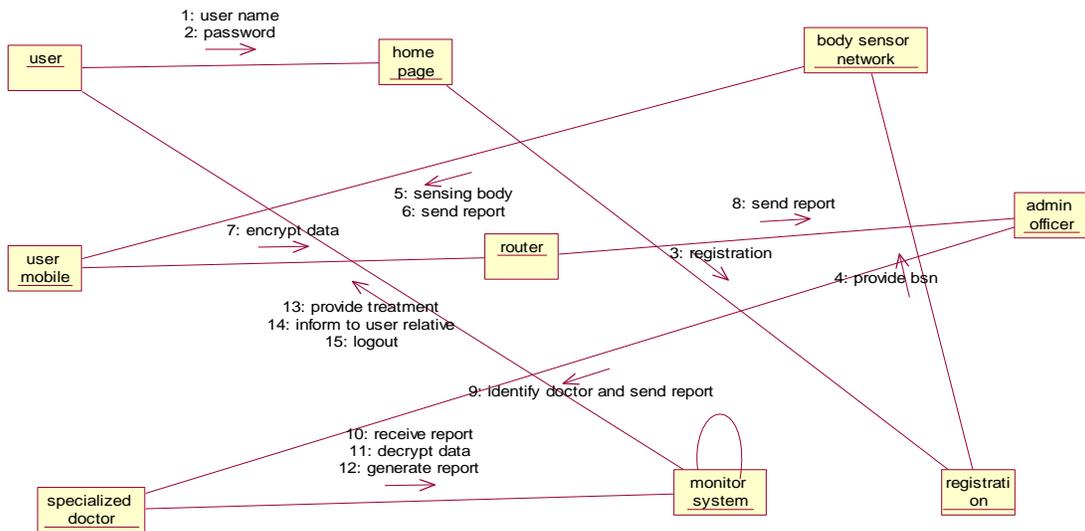
USECASE DIAGRAM



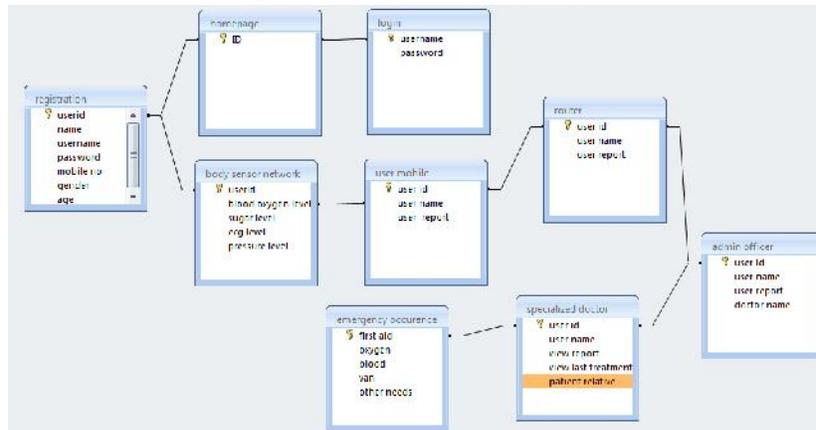
SEQUENCE DIAGRAM



COLLABORATION DIAGRAM



IMPLEMENTATION STEPS OF THE PROJECT
IMPLEMENTATION STEPS



**SAMPLE CODING
SERVER**

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Net.Sockets;
using System.Net;
using System.IO;

namespace server
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            FolderBrowserDialog f = new FolderBrowserDialog();
            f.ShowDialog();
            if (f.SelectedPath != "")
            {
                DestCode.receivedPath = f.SelectedPath;
                label5.Text = f.SelectedPath;
            }
            else
            {
                MessageBox.Show("Please Select a File Receiving Path.\r\n Else You Can not Receive the File");
            }
        }
        DestCode d = new DestCode();
        private void Form1_Load(object sender, EventArgs e)
        {
            backgroundWorker1.RunWorkerAsync();
        }

        private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
        {
            d.StartServer();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            if (DestCode.receivedmsg == "Received")
            {
                DestCode.receivedmsg = "";
                listBox1.Items.Add(DestCode.curMsg);
            }
            else if (DestCode.receivedmsg == "Save")

```

```

{
    DestCode.receivedmsg = "";
    lblRes.Text = "File Saved";
}
else
{
    lblRes.Text = "";
}
}
}
class DestCode
{
    IPEndPoint ipEnd;
    Socket sock;
    public DestCode()
    {
        IPEndPoint ipEntry = Dns.GetHostEntry(Environment.MachineName);
        IPAddress IpAddr = ipEntry.AddressList[0];
        ipEnd = new IPEndPoint(IpAddr, 5003);
        sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.IP);
        sock.Bind(ipEnd);
    }
    public static string receivedmsg;
    public static string receivedPath;
    public static string curMsg = "";
    byte[] write;
    byte[] data1;
    byte[] data2;
    byte[] data3;
    byte[] data4;
    byte[] data5;
    int i = 0;
    int len = 0;
    public void StartServer()
    {
        try
        {
            //curMsg = "Starting...";
            sock.Listen(100);

            // curMsg = "Running and waiting to receive file.";
            Socket clientSock = sock.Accept();

            byte[] clientData = new byte[1024 * 5000];

            int receivedBytesLen = clientSock.Receive(clientData);
            curMsg = Convert.ToString(receivedBytesLen) + " Bits Received ";
            receivedmsg = "Received";
            if (i == 0)
            {
                data1 = new byte[receivedBytesLen];
                Array.Copy(clientData, data1, receivedBytesLen);
                len = receivedBytesLen;
                i++;
                //receivedmsg = "";
            }
        }
    }
}

```

```

else if (i == 1)
{

    len += receivedBytesLen;
    data2 = new byte[receivedBytesLen];
    Array.Copy(clientData, data2, receivedBytesLen);
    i++;
}
else if (i == 2)
{
    len += receivedBytesLen;
    data3 = new byte[receivedBytesLen];
    Array.Copy(clientData, data3, receivedBytesLen);
    i++;
}
else if (i == 3)
{
    len += receivedBytesLen;
    data4 = new byte[receivedBytesLen];
    Array.Copy(clientData, data4, receivedBytesLen);
    i++;

    Application.DoEvents();
    write = new byte[len];
}

if (i == 4)
{
    //byte[] write = new byte[data1.Length + data2.Length + data3.Length + data4.Length];
    if (receivedPath == null )
    {
        MessageBox.Show(" Path was Not selected for Save the
File. ", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Hand );
    }
    else
    {

        Array.Copy(data1, write, data1.Length);
        Array.Copy(data2, 0, write, data1.Length + 1, data2.Length);
        Array.Copy(data3, 0, write, data1.Length + data2.Length, data3.Length);
        Array.Copy(data4, 0, write, data1.Length + data2.Length + data3.Length, data4.Length);

        int fileNameLen = BitConverter.ToInt32(write, 0);
        string fileName = Encoding.ASCII.GetString(write, 4, fileNameLen);

        BinaryWriter bWrite = new BinaryWriter(File.Open(receivedPath + "/" + fileName, FileMode.Append)); ;
        bWrite.Write(write, 4 + fileNameLen, write.Length - 4 - fileNameLen);

        //curMsg = "Saving file...";

        bWrite.Close();
        curMsg = "";

        data1 = null;
        data2 = null;

```



```

byte[] fileNameLen;

private void Form2_Load(object sender, EventArgs e)

{
    button2.Enabled = false ;
    button3.Enabled = false;
}

private void button1_Click(object sender, EventArgs e)
{
    data1 = null;
    data2 = null;
    data3 = null;
    data4 = null;
    textBox1.Text = "";
    openFileDialog1.ShowDialog();
    textBox1.Text = openFileDialog1.FileName;
    fileDes = openFileDialog1.FileName;

    if (fileDes == "openFileDialog1")
    {
        textBox1.Text = "";
        MessageBox.Show("Please Select the file");
    }
    else
    {
        button2.Enabled = true;
        len = fileDes.Length;
        fileini = fileDes.Substring(fileDes.IndexOf("\\") + 1);

        filePath = "";

        fileDes = fileDes.Replace("\\", "/");
        while (fileDes.IndexOf("/") > -1)
        {
            filePath += fileDes.Substring(0, fileDes.IndexOf("/") + 1);
            fileDes = fileDes.Substring(fileDes.IndexOf("/") + 1);
        }

        fileNameByte = Encoding.ASCII.GetBytes(fileDes);

        fileData = File.ReadAllBytes(filePath + fileDes);
        clientData = new byte[4 + fileNameByte.Length + fileData.Length];
        fileNameLen = BitConverter.GetBytes(fileNameByte.Length);

        fileNameLen.CopyTo(clientData, 0);
        fileNameByte.CopyTo(clientData, 4);
        fileData.CopyTo(clientData, 4 + fileNameByte.Length);
        int psize = clientData.Length / 1024;
        lblfilesize.Text = psize.ToString() + " KB";

        button1.Enabled = false;
    }
}

```

```

}

private void button2_Click(object sender, EventArgs e)
{
    //btnSend.Enabled = true;
    button3.Enabled = true;
    data2 = new byte[800];
    data1 = new byte[500];
    data3 = new byte[3000];
    data4 = new byte[clientData.Length - 4300];
    Array.Copy(clientData, data1, 500);
    Array.Copy(clientData, 501, data2, 0, 800);
    Array.Copy(clientData, 1300, data3, 0, 3000);
    Array.Copy(clientData, 4300, data4, 0, clientData.Length - 4300);

    listBox1.Items.Add("" + data1.Length + " Bits");
    listBox1.Items.Add("" + data2.Length + " Bits");
    listBox1.Items.Add("" + data3.Length + " Bits");
    listBox1.Items.Add("" + data4.Length + " Bits");

    button2.Enabled = false;
}

private void button3_Click(object sender, EventArgs e)
{
    //timer1.Enabled = true;
    button3.Enabled = false;
    button1.Enabled = true;
    if (data1.Length < 1000)
    {
        send(5001, txtIp1.Text, data1);
        System.Threading.Thread.Sleep(3000);
    }
    else
    {
        send(5002,txtIp2.Text, data1);
    }
    if (data2.Length < 1000)
    {
        send(5001, txtIp1.Text, data2);
        System.Threading.Thread.Sleep(3000);
    }
    else
    {
        send(5002, txtIp2.Text, data1);
    }
    if (data3.Length >= 1000)
    {
        send(5002, txtIp2.Text, data3);
        System.Threading.Thread.Sleep(3000);
    }
    else
    {
        send(5001, txtIp1.Text, data3);
    }
}

```

```

if (data4.Length >= 1000)
    {
        send(5002, txtIp2.Text, data4);
        System.Threading.Thread.Sleep(3000);
    }
else
    {
        send(5001, txtIp1.Text, data4);
    }
}
public void send( int p, string ips, byte[] cdata)
{
    timer1.Enabled = false;
    port = p;
    try
    {
        IPAddress[] ipAddress = Dns.GetHostAddresses(ips);
        IPEndPoint ipEnd = new IPEndPoint(ipAddress[0], port);
        Socket clientSock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.IP);

        clientSock.Connect(ipEnd);
        System.Threading.Thread.Sleep(1000);
        clientSock.Send(cdata);
        clientSock.Close();
    }
    catch (Exception ex)
    {
        if (ex.Message == "A connection attempt failed because the connected party did not properly respond after a period of
time, or established connection failed because connected host has failed to respond")
            {
                }
            else
            {
                if (ex.Message == "No connection could be made because the target machine actively refused it")
                    {
                        }
                    else
                    {
                        }
                }
            }
        }
    timer1.Enabled = true;
}

private void button4_Click(object sender, EventArgs e)
{
    this.Close();
}

private void lblfilesize_Click(object sender, EventArgs e)
{
}
}
}

```

TESTING STAGES OF THE PROJECT

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product.

It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application it is done after the completion of an individual unit before integration.

This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software

components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

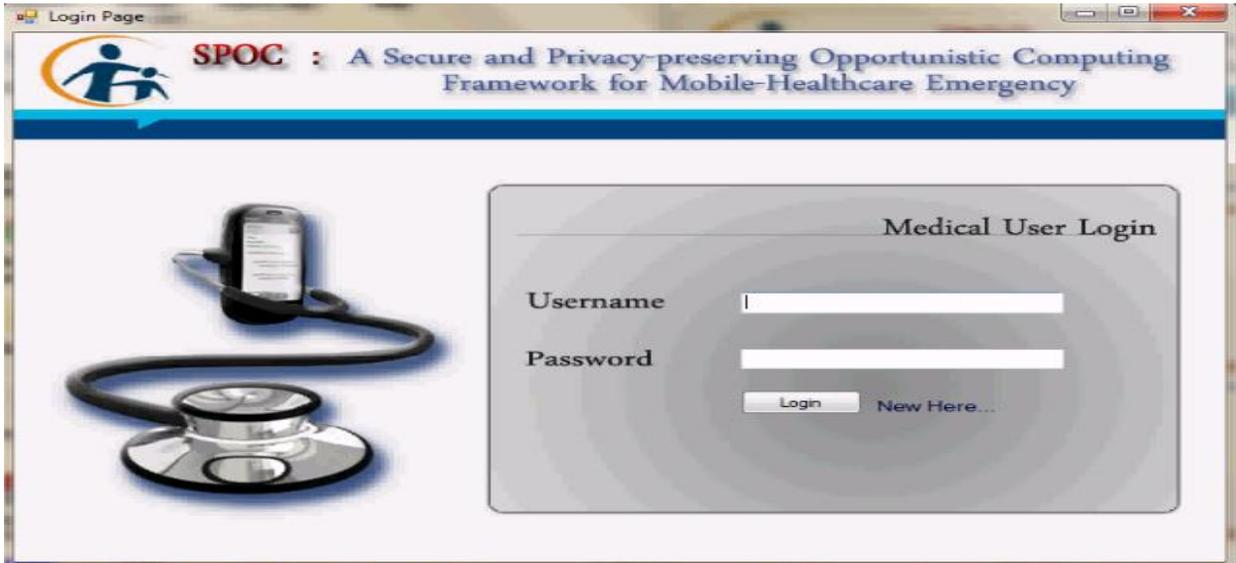
Test Results: All the test cases mentioned above passed successfully. No defects encountered.

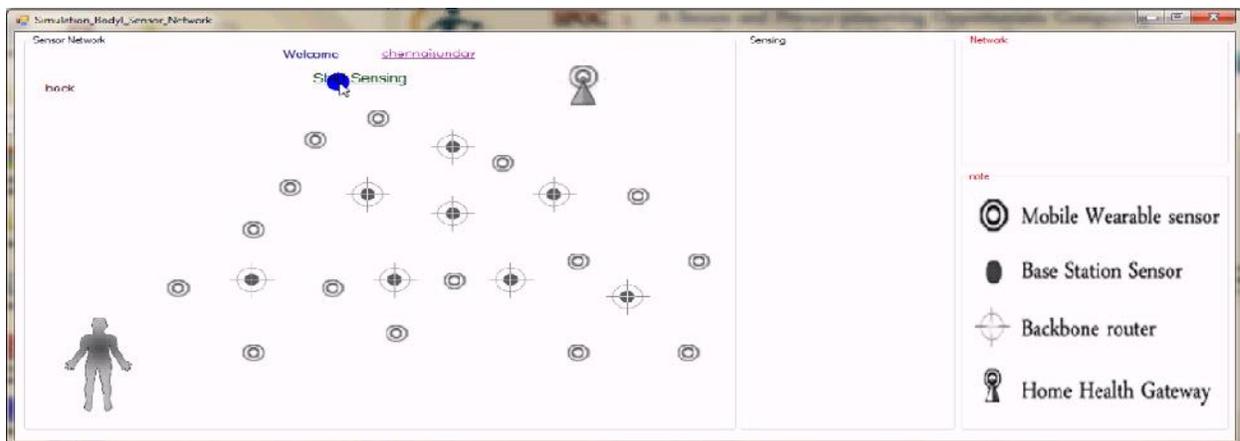
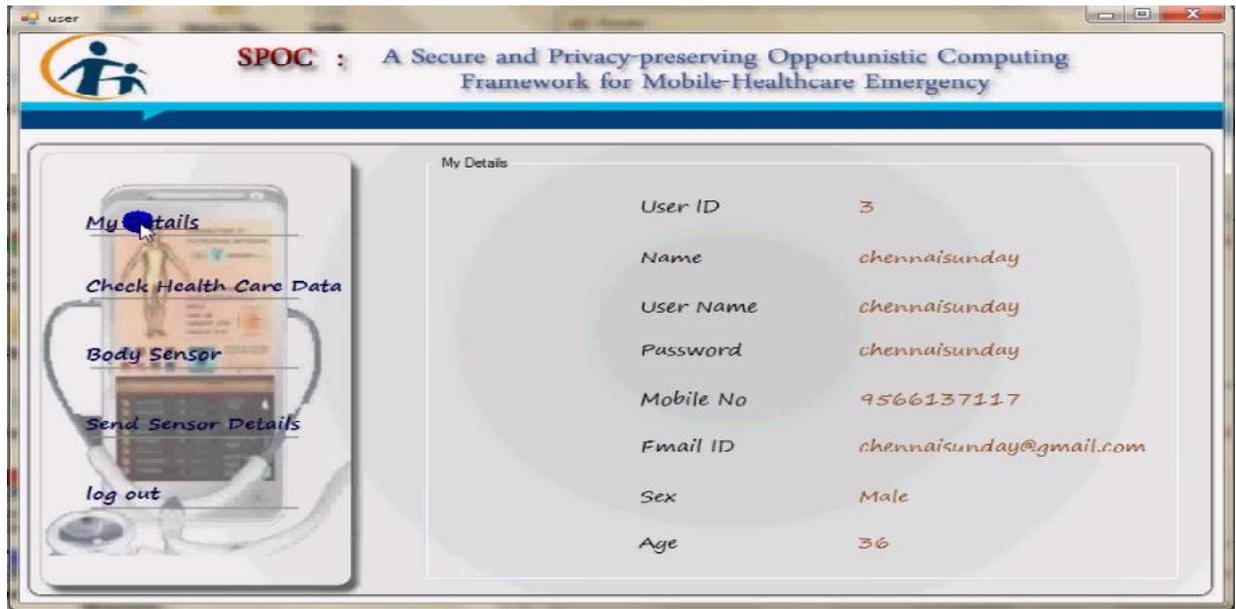
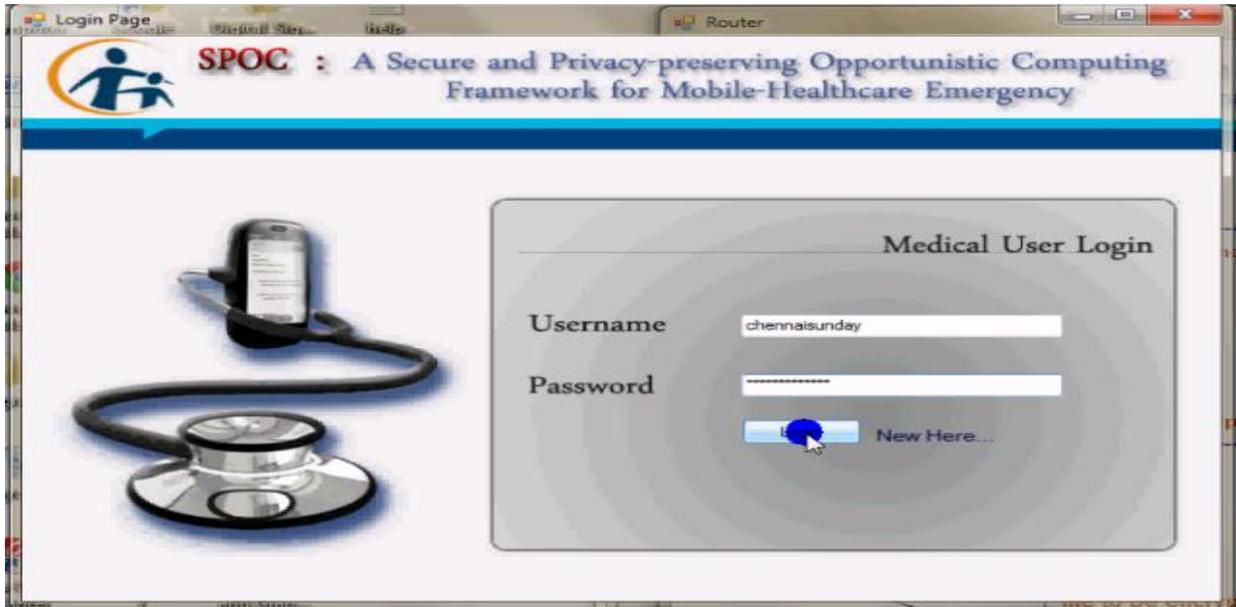
Acceptance Testing

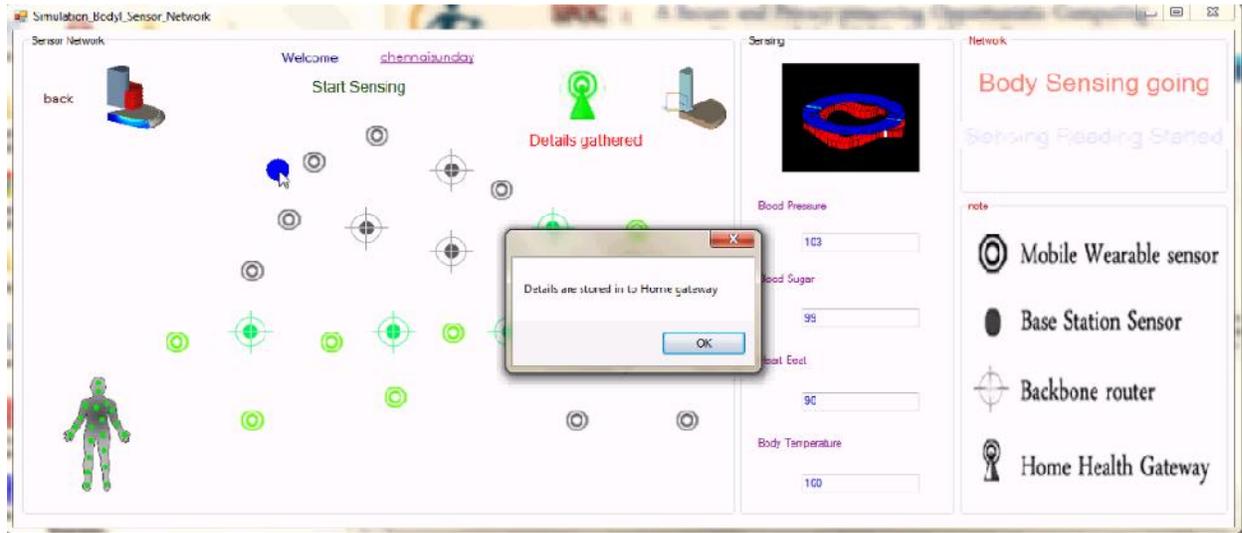
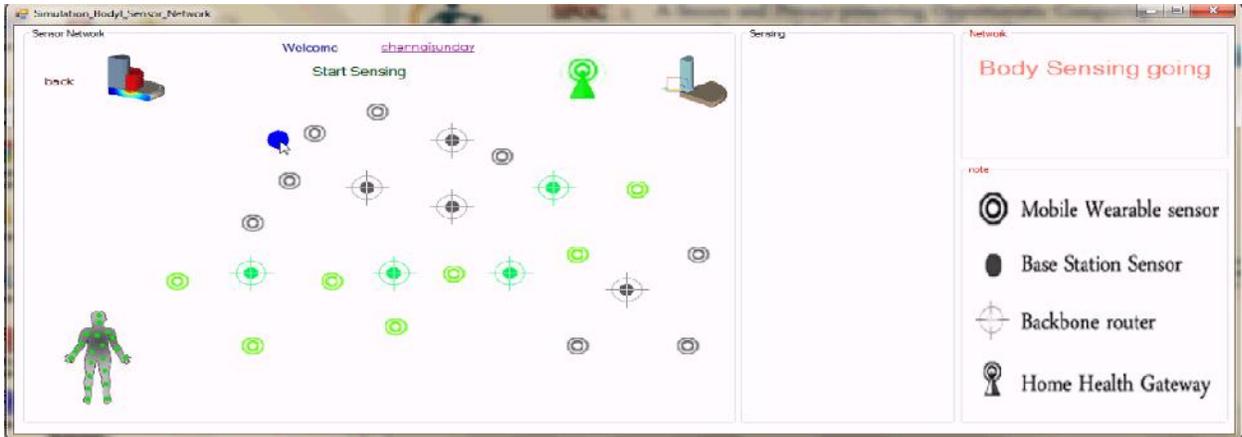
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

INPUT/OUTPUT SCREENS







Form1

 **SPOC** : A Secure and Privacy-preserving Opportunistic Computing Framework for Mobile-Healthcare Emergency

Health care User Login



Medical ID

Medical Key

Medicaluser

 **SPOC** : A Secure and Privacy-preserving Opportunistic Computing Framework for Mobile-Healthcare Emergency



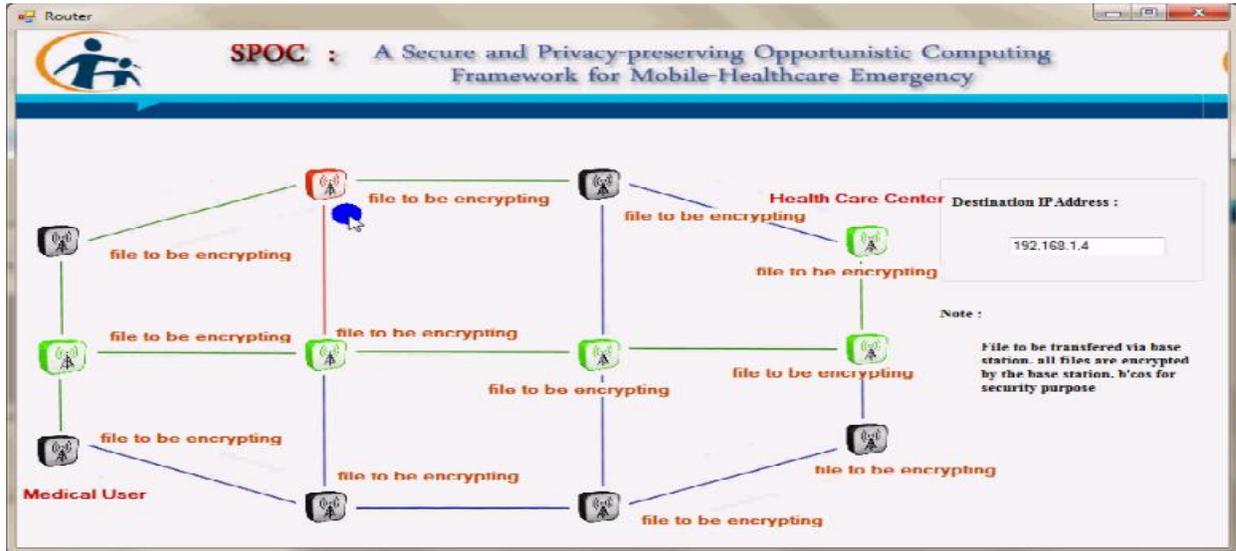
senddetails

 **SPOC** : A Secure and Privacy-preserving Opportunistic Computing Framework for Mobile-Healthcare Emergency

back Welcome chennaisunday

Send	Name	Mobile	Email	Sex	Age	Blind Pressure
Send details	chennaisunday	9566137117	chennaisunday@...	Male	36	103

File transferred



SPOC : A Secure and Privacy-preserving Opportunistic Computing Framework for Mobile-Healthcare Emergency

File Received

back

The interface displays a list of received files, each represented by a small icon and a label "file decrypting and store in database". Below the list, there is a button labeled "open the database list".

SPOC : A Secure and Privacy-preserving Opportunistic Computing Framework for Mobile-Healthcare Emergency

User ID	User Name	Mobile No	Email ID	Sex	Age	Blood Pressure Level
1	Kingslin Arunkumar	9568315796	king@gmail.com	Male	26	113
3	chennaisunday	9566137117	chennaisunday@...	Male	36	103

Emergency

SPOC : A Secure and Privacy-preserving Opportunistic Computing Framework for Mobile-Healthcare Emergency

Patient Details

User Id: User Name:

	User ID	User Name	Mobile No	Mail	Sex	Age	Blood Pressure
▶	3	chennaisunday	9566137117	chennaisunday@...	Male	36	103
*							

report

Main Report

Current Page No.: 1 Total Page No.: 1+ Zoom Factor: 100%

report

Main Report

Health care report

Kinglin Anolkumar 100%

Nishal Devi 0%

chennaisunday 0%

Test: 100%

6/20/2011

00 Kinglin 9,566,315,796.00 king@gmail.com Male 6.00 3.00 39.00 12.00 June

EVALUATION REPORTS OF THE PROJECT

CONCLUSION

In this paper, we have proposed a secure and privacy preserving opportunistic computing (SPOC) framework for m-Healthcare emergency, which mainly exploits how to use opportunistic computing to achieve high reliability of PHI process and transmission in emergency while minimizing the privacy disclosure during the opportunistic computing. Detailed security analysis shows that the proposed SPOC framework can achieve the efficient user-centric privacy access control. In addition, through extensive performance evaluation, we have also demonstrated the proposed SPOC framework can balance the high-intensive PHI process and transmission and minimizing the PHI privacy disclosure in m-Healthcare emergency.

In our future work, we intend to carry on smart phone based experiments to further verify the effectiveness of the proposed SPOC framework. In addition, we will also exploit the security issues of PPSPC with internal attackers, where the internal attackers will not honestly follow the protocol.

REFERENCES

- [1] A. Toninelli, R. Montanari, and A. Corradi, “Enabling Secure Service Discovery in Mobile Healthcare Enterprise Networks,” *IEEE Wireless Comm.*, vol. 16, no. 3, pp. 24-32, June 2009.
- [2] R. Lu, X. Lin, X. Liang, and X. Shen, “Secure Handshake with Symptoms-Matching: The Essential to the Success of M healthcare Social Network,” *Proc. Fifth Int’l Conf. Body Area Networks (Body Nets ’10)*, 2010.
- [3] Y. Ren, R.W.N. Pazzi, and A. Boukerche, “Monitoring Patients via a Secure and Mobile Healthcare System,” *IEEE Wireless Comm.*, vol. 17, no. 1, pp. 59-65, Feb. 2010.
- [4] R. Lu, X. Lin, X. Liang, and X. Shen, “A Secure Handshake Scheme with Symptoms-Matching for m Healthcare Social Network,” *Mobile Networks and Applications—special issue on wireless and personal comm.*, vol. 16, no. 6, pp. 683-694, 2011.
- [5] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, “Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption,” *IEEE Trans. Parallel and Distributed System*, to be published.