**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TABLE OF CONTENTS** | **PAGE NO.** |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.    10. | ABSTRACT  INTRODUCTION   * 1. . LITERATURE SURVEY   2. . MODULES DESCRIPTION   STUDY OF THE SYSTEM  2.1. FEASABILITY STUDY  2.2. EXISTING SYSTEM  2.3. PROPOSED SYSTEM  SYSTEM SPECIFICATION  3.1. HARDWARE REQUIREMENTS  3.2. SOFTWARE REQUIREMENTS  SYSTEM DEVELOPEMENT ENVIRONMENT   * 1. . INTRODUCTION TO JAVA   2. . SERVLETS/ JSP   3. . JDBC   SYSTEM DESIGN & DEVELOPMENT   1. . INTRODUCTION 2. . NORMALIZATION 3. . E-R DIAGRAM 4. . DATA DICTIONARY 5. . DATA FLOW DIAGRAM 6. . ACTIVITY DIAGRAM 7. . USE CASE DIAGRAM 8. . SEQUENCE DIAGRAM 9. . CLASS DIAGRAM   SAMPLE SOURCE CODE  SCREEN LAYOUT  SYSTEM TESTING AND IMPLEMENTATION  CONCLUSION  BIBLIOGRAPHY | iii  1  9  13  14  38  62  68  78 |

**Efficient Authentication for Mobile and Pervasive Computing**

**Abstract:**

With today’s technology, many applications rely on the existence of small devices that can exchange information and form communication networks. In a significant portion of such applications, the confidentiality and integrity of the communicated messages are of particular interest. In this work, we propose two novel techniques for authenticating short encrypted messages that are directed to meet the requirements of mobile and pervasive applications. By taking advantage of the fact that the message to be authenticated must also be encrypted, we propose provably secure authentication codes that are more efficient than any message authentication code in the literature. The key idea behind the proposed techniques is to utilize the security that the encryption algorithm can provide to design more efficient authentication mechanisms, as opposed to using standalone authentication primitives.

**INTRODUCTION**

PRESERVING the integrity of messages exchanged over public channels is one of the classic goals in cryptography and the literature is rich with message authentication code (MAC) algorithms that are designed for the sole purpose of preserving message integrity. Based on their security, MACs can be either unconditionally or computationally secure. Unconditionally secure MACs provide message integrity against forgers with unlimited computational power. On the other hand, computationally secure MACs are only secure when forgers have limited computational power.

A popular class of unconditionally secure authentication is based on universal hash-function families, pioneered by Carter and Wegman. Since then, the study of unconditionally secure message authentication based on universal hash functions has been attracting research attention, both from the design and analysis standpoints. The basic concept allowing for unconditional security is that the authentication key can only be used to authenticate a limited number of exchanged messages. Since the management of one-time keys is considered impractical in many applications, computationally secure MACs have become the method of choice for most real-life applications. In computationally secure MACs, keys can be used to authenticate an arbitrary number of messages. That is, after agreeing on a key, legitimate users can exchange an arbitrary number of authenticated messages with the same key. Depending on the main building block used to construct them, computationally secure MACs can be classified into three main categories: block cipher based, cryptographic hash function based, or universal hash-function family based.

CBC-MAC is one of the most known block cipher based MACs, specified in the Federal Information Processing Standards publication 113 and the International Organization for Standardization ISO/IEC 9797-1. CMAC, a modified version of CBC-MAC, is presented in the NIST special publication 800-38B, which was based on the OMAC of. Other block cipher based MACs include, but are not limited to, XOR-MAC and PMAC. The security of different MACs has been exhaustively studied.

The use of one-way cryptographic hash functions for message authentication was introduced by Tsudik. A popular example of the use of iterated cryptographic hash functions in the design of message authentication codes is HMAC, which was proposed by Bellare. HMAC was later adopted as a standard. Another cryptographic hash function based MAC is the MDx-MAC proposed by Preneel and Oorschot. HMAC and two variants of MDx- MAC are specified in the International Organization for Standardization ISO/IEC 9797-2. Bosselaers et al. described how cryptographic hash functions can be carefully coded to take advantage of the structure of the Pentium processor to speed up the authentication process.

The use of universal hash-function families in the Carter-Wegman style is not restricted to the design of unconditionally secure authentication. Computationally secure MACs based on universal hash functions can be constructed with two rounds of computations. In the first round, the message to be authenticated

is compressed using a universal hash function. Then, in the second round, the compressed image is processed with a cryptographic function (typically a pseudorandom function1). Popular examples of computationally secure universal hashing based MACs include, but are not limited to.

Indeed, universal hashing based MACs give better performance when compared to block cipher or cryptographic hashing based MACs. In fact, the fastest MACs in the cryp-tographic literature are based on universal hashing. The main reason behind the performance advantage of universal hashing based MACs is the fact that processing messages block by block using universal hash functions is orders of magnitude faster than processing them block by block using block ciphers or cryptographic hash functions.

One of the main differences between unconditionally secure MACs based on universal hashing and computationally secure MACs based on universal hashing is the requirement to process the compressed image with a cryptographic primitive in the latter class of MACs. This round of computation is necessary to protect the secret key of the universal hash function. That is, since universal hash functions are not cryptographic functions, the observation of multiple message-image pairs can

reveal the value of the hashing key. Since the hashing key is used repeatedly in computationally secure MACs, the exposure of the hashing key will lead to breaking the security of the MAC. Thus, processing the compressed image with a cryptographic primitive is necessary for the security of this class of MACs. This implies that unconditionally secure MACs based on universal hashing are more efficient than computationally secure ones. On the negative side, unconditionally secure universal hashing based MACs are considered impractical in most modern applications, due to the difficulty of managing one-time keys.

There are two important observations to make about existing MAC algorithms. First, they are designed independently of any other operations required to be performed on the message to be authenticated. For instance, if the authenticated message must also be encrypted, existing MACs are not designed to utilize the functionality that can be provided by the underlying encryption algorithm. Second, most existing MACs are designed for the general computer communication systems, independently of the properties that messages can possess. For example, one can find that most existing MACs are inefficient when the messages to be authenticated are short. (For instance, UMAC, the fastest reported message authentication code in the cryptographic literature, has undergone large algorithmic changes to increase its speed on short messages.)

Nowadays, however, there is an increasing demand for the deployment of networks consisting of a collection of small devices. In many practical applications, the main purpose of such devices is to communicate short messages. A sensor network, for example, can be deployed to monitor certain events and report some collected data. In many sensor network applications, reported data consist of short confidential measurements. Consider, for instance, a sensor network deployed in a battlefield with the purpose of reporting the existence of moving targets or other temporal activities. In such applications, the confidentiality and integrity of reported events are of critical importance.

In another application, consider the increasingly spreading deployment of radio frequency identification (RFID) systems. In such systems, RFID tags need to identify themselves to authorized RFID readers in an authenticated way that also preserves their privacy. In such scenarios, RFID tags usually encrypt their identity, which is typically a short string (for example, tags unique identifiers are 64-bit long in the EPC Class-1 Generation-2 standard [39]), to protect their privacy. Since the RFID reader must also authenticate the identity of the RFID tag, RFID tags must be equipped with a message authentication mechanism.

Another application that is becoming increasingly important is the deployment of body sensor networks. In such applications, small sensors can be embedded in the patient’s body to report some vital signs. Again, in some applications the confidentiality and integrity of such reported messages can be important.

There have been significant efforts devoted to the design of hardware efficient implementations that suite such small devices. For instance, hardware efficient implementations of block ciphers have been proposed. Implementations of hardware efficient cryptographic hash functions have also been proposed. However, there has been little or no effort in the design of special algorithms that can be used for the design of message authentication codes that can utilize other operations and the special properties of such networks. In this paper, we provide the first such work.

* 1. **LITERATURE SURVEY:**

PRESERVING the integrity of messages exchanged over public channels is one of the classic goals in cryptography and the literature is rich with message authentication code (MAC) algorithms that are designed for the sole purpose of preserving message integrity. Based on their security, MACs can be either unconditionally or computationally secure. Unconditionally

secure MACs provide message integrity against forgers with unlimited computational power. On the other hand, computationally secure MACs are only secure when forgers have limited computational power. A popular class of unconditionally secure authentication is based on universal hash-function families, pioneered by Carter and Wegman. Since then, the study of unconditionally secure message authentication based on universal hash functions has been attracting research attention, both from the design and analysis standpoints. The basic concept allowing for unconditional security is that the authentication key can only be used to authenticate a limited number of exchanged messages. Since the management of one-time keys is considered impractical in many applications, computationally secure MACs have become the method of choice for most real-life applications. In computationally secure MACs, keys can be used to authenticate an arbitrary number of messages. That is, after agreeing on a key, legitimate users can exchange an arbitrary number of authenticated messages with the same key. Depending on the main building block used to construct them, computationally secure MACs can be classified into three main categories: block cipher based, cryptographic hash function based, or universal hash-function family based. CBC-MAC is one of the most known block cipher based MACs, specified in the Federal Information Processing Standards publication and the International Organization for Standardization CMAC, a modified version of CBC-MAC, is presented in the NIST , which was based on the OMAC. Other block cipher based MACs include, but are not limited to, XOR-MAC and PMAC. The security of different MACs has been exhaustively studied. The use of one-way cryptographic hash functions for message authentication was introduced by Tsudik. A popular example of the use of iterated cryptographic hash functions in the design of message authentication codes is HMAC, which was proposed by Bellare et al. in HMAC was later adopted as a standard. Another cryptographic hash function based MAC is the MDx-MAC proposed by Preneel and Oorschot . HMAC and two variants of MDx- MAC are specified in the International Organization for Standardization ISO/IEC. Bosselaers et al. described how cryptographic hash functions can be carefully coded to take advantage of the structure of the Pentium processor to speed up the authentication process. The use of universal hash-function families in the Carter- Wegman style is not restricted to the design of unconditionally secure authentication. Computationally secure MACs based on universal hash functions can be constructed with two rounds of computations. In the first round, the message to be authenticated is compressed using a universal hash function. Then, in the second round, the compressed image is processed with a cryptographic function (typically a pseudorandom function. Popular examples of computationally secure universal hashing based MACs include, but are not limited. Indeed, universal hashing based MACs give better performance when compared to block cipher or cryptographic hashing based MACs. In fact, the fastest MACs in the cryp-tographic literature are based on universal hashing. The main reason behind the performance advantage of universal hashing based MACs is the fact that processing messages block by block using universal hash functions is orders of magnitude faster than processing them block by block using block ciphers or cryptographic hash functions.

**1.2 Implementation modules:**

1. Authenticating Short Encrypted Messages
2. Security Model
3. Data Privacy
4. Security of the Authenticated Encryption Composition

**AUTHENTICATING SHORT ENCRYPTED MESSAGES:**

In this module, we describe our first authentication scheme that can be used with any IND-CPA secure encryption algorithm .An important assumption we make is that messages to be authenticated are no longer than a predefined length. This includes applications in which messages are of fixed length that is known a priori, such as RFID systems in which tags need to authenticate their identifiers, sensor nodes reporting events that belong to certain domain or measurements within a certain range, etc. The novelty of the proposed scheme is to utilize the encryption algorithm to deliver a random string and use it to reach the simplicity and efficiency of one-time pad authentication without the need to manage impractically long keys.

**Security Model:**

A message authentication scheme consists of a signing algorithm S and a verifying algorithm V. The signing algorithm might be probabilistic, while the verifying one is usually not. Associated with the scheme are parameters and N describing the length of the shared key and the resulting authentication tag, respectively.

**Security of the Authenticated Encryption Composition:**

In this module, it defined two notions of integrity for authenticated encryption systems: the first is integrity of plaintext (INT-PTXT) and the second is integrity of cipher text (INT-CTXT). Combined with encryption algorithms that provide in-distinguish ability under chosen plaintext attacks(IND-CPA), the security of different methods for constructing generic compositions is analyzed. Note that our construction is an instance of the Encrypt-and-Authenticate (E&A) generic composition since the plaintext message goes to the encryption algorithm as an input, and the same plaintext message goes to the authentication algorithm as an input.

**Data Privacy:**

Recall that two pieces of information are transmitted to the intended receiver (the cipher text and the authentication tag),both of which are functions of the private plaintext message. Now, when it comes to the authentication tag, observe that then once r serves as a one-time key (similar to the role r plays in the construction of Section. The formal analysis that the authentication tag does not compromise message privacy is the same as the one provided . The cipher text of equation ,on the other hand, is a standard CBC encryption and its security is well-studied; thus, we give the theorem statement below without a formal proof (interested readers may refer to textbooks in cryptography.

**2. SYSTEM STUDY**

**2.1 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**2.2 Existing System:**

There are two important observations to make about existing MAC algorithms. First, they are designed independently of any other operations required to be performed on the message to be authenticated. For instance, if the authenticated message must also be encrypted, existing MACs are not designed to utilize the functionality that can be provided by the underlying encryption algorithm. Second, most existing MACs are designed for the general computer communication systems, independently of the properties that messages can possess. For example, one can find that most existing MACs are inefficient when the messages to be authenticated are short. (For instance, UMAC, the fastest reported message authentication code in the cryptographic literature, has undergone large algorithmic changes to increase its speed on short messages ).

**Disadvantages:**

1.Existing MACs are not designed to utilize the functionality that can be provided by the underlying encryption algorithm.

2.Most existing MACs are designed for the general computer communication systems, independently of the properties that messages can possess.

**2.3 Proposed System:**

We propose the following research question: if there is an application in which messages that need to be exchanged are short and both their privacy and integrity need to be preserved, can one do better than simply encrypting the messages using an encryption algorithm and authenticating them using standard MAC algorithm? We answer the question by proposing two new techniques for authenticating short encrypted messages that are more efficient than existing approaches. In the first technique, we utilize the fact that the message to be authenticated is also encrypted, with any secure encryption algorithm, to append a short random string to be used in the authentication process.

**Advantages:**

**1**.More security, using two concepts one is mobile computing and another one is pervasive computing.

**2**.Therandom strings used for different operations are independent, the authentication algorithm can benefit from the simplicity of unconditional secure authentication to allow for faster and more efficient authentication, without the difficulty to manage one-time keys. In the second technique, we make the extra assumption that the used encryption algorithm is block cipher based to further improve the computational efficiency of the first technique.

3. SYSTEM SPECIFICATION

3.1. HARDWARE REQUIREMENTS:

# Processor - Pentium –III

Speed - 1.1 Ghz

RAM - 256 MB(min)

Hard Disk - 20 GB

Floppy Drive - 1.44 MB

Key Board - Standard Windows Keyboard

Mouse - Two or Three Button Mouse

Monitor - SVGA

# 3.2 S/W System Configuration:-

* Operating System :Windows95/98/2000/XP
* Front End : java, jdk1.6
* Database : My sqlserver 2005
* Database Connectivity : JDBC.

4. SYSTEM DEVELOPEMENT ENVIRONMENT

# 4.1. INTRODUCTION TO JAVA:

## *Java Technology*

Java technology is both a programming language and a platform.

### The Java Programming Language

### The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

* + - Simple
    - Architecture neutral
    - Object oriented
    - Portable
    - Distributed
    - High performance
    - Interpreted
    - Multithreaded
    - Robust
    - Dynamic
    - Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



### The Java Platform

A platform is the hardware or software environment in which a program runs. We’ve already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

* The Java Virtual Machine (Java VM)
* The Java Application Programming Interface (Java API)

You’ve already been introduced to the Java VM. It’s the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that’s running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

## What Can Java Technology Do?

The most common types of programs written in the Java programming language are applets and applications. If you’ve surfed the Web, you’re probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

* **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
* **Applets**: The set of conventions used by applets.
* **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
* **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
* **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
* **Software components**: Known as JavaBeansTM, can plug into existing component architectures.
* **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
* **Java Database Connectivity (JDBCTM)**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



## How Will Java Technology Change My Life?

We can’t promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

* **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it’s easy to learn, especially for programmers already familiar with C or C++.
* **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
* **Write better code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people’s tested code and introduce fewer bugs.
* **Develop programs more quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
* **Avoid platform dependencies with 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure JavaTM Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
* **Write once, run anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
* **Distribute software more easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded “on the fly,” without recompiling the entire program.

### ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn’t change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn’t as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

**JDBC**

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

### JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. ***SQL Level API***

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

1. ***SQL Conformance***

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

1. ***JDBC must be implemental on top of common database interfaces***The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.
2. ***Provide a Java interface that is consistent with the rest of the Java system***

Because of Java’s acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

1. ***Keep it simple***

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

1. ***Use strong, static typing wherever possible***

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

1. ***Keep the common cases simple***

Because more often than not, the usual SQL calls used by the programmer are simple SELECT’s, INSERT’s, DELETE’s and UPDATE’s, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

Finally we decided to proceed the implementation using Java Networking.

And for dynamically updating the cache table we go for MS Access database.

Java ha two things: a programming language and a platform.

Java is a high-level programming language that is all of the following

Simple Architecture-neutral

Object-oriented Portable

Distributed High-performance

Interpreted multithreaded

Robust Dynamic

Secure

Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.

**Java Program**

**Compilers**

**Interpreter**

**My Program**

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make “write once, run anywhere” possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

## Networking

### TCP/IP stack

The TCP/IP stack is shorter than the OSI one:



TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

### IP datagram’s

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

### UDP

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

### TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

### Internet addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

### Network address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

### Subnet address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

### Host address

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

### Total address



The 32 bit address is usually written as 4 integers separated by dots.

### Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

### Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call socket. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write File functions.

#include <sys/types.h>

#include <sys/socket.h>

int socket(int family, int type, int protocol);

Here "family" will be AF\_INET for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

**JFree Chart**

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types;

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

JFreeChart is "open source" or, more specifically, [free software](http://www.gnu.org/philosophy/free-sw.html). It is distributed under the terms of the [GNU Lesser General Public Licence](http://www.gnu.org/licenses/lgpl.html) (LGPL), which permits use in proprietary applications.

## 1. Map Visualizations

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFreeChart;

Testing, documenting, testing some more, documenting some more.

## 2. Time Series Chart Interactivity

Implement a new (to JFreeChart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

## 3. Dashboards

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFreeChart chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

## 4. Property Editors

The property editor mechanism in JFreeChart only handles a small subset of the properties that can be set for charts. Extend (or reimplement) this mechanism to provide greater end-user control over the appearance of the charts.

**J2ME (Java 2 Micro edition):-**

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems." Announced in June 1999 at the JavaOne Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices.

**1. General J2ME architecture**



J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes. The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. We'll discuss configurations in detail in the The profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. We'll cover profiles in depth in the The following graphic depicts the relationship between the different virtual machines, configurations, and profiles. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine -- it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.

**2.Developing J2ME applications**

Introduction In this section, we will go over some considerations you need to keep in mind when developing applications for smaller devices. We'll take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, we'll explore packaging and deployment and the role preverification plays in this process.

**3.Design considerations for small devices**

Developing applications for small devices requires you to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before you begin coding. Correcting the code because you failed to consider all of the "gotchas" before developing the application can be a painful process. Here are some design strategies to consider:

\* Keep it simple. Remove unnecessary features, possibly making those features a separate, secondary application.

\* Smaller is better. This consideration should be a "no brainer" for all developers. Smaller applications use less memory on the device and require shorter installation times. Consider packaging your Java applications as compressed Java Archive (jar) files.

\* Minimize run-time memory use. To minimize the amount of memory used at run time, use scalar types in place of object types. Also, do not depend on the garbage collector. You should manage the memory efficiently yourself by setting object references to null when you are finished with them. Another way to reduce run-time memory is to use lazy instantiation, only allocating objects on an as-needed basis. Other ways of reducing overall and peak memory use on small devices are to release resources quickly, reuse objects, and avoid exceptions.

**4.Configurations overview**

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, though others may be defined in the future:

\* **Connected Limited Device Configuration (CLDC)** is used specifically with the KVM for 16-bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications. Its size limitations make CLDC more interesting and challenging (from a development point of view) than CDC. CLDC is also the configuration that we will use for developing our drawing tool application. An example of a small wireless device running small applications is a Palm hand-held computer.

\* **Connected Device Configuration (CDC)** is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.

**5.J2ME profiles**

**What is a J2ME profile?**

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: KJava and MIDP. Both KJava and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

A skeleton profile upon which you can create your own profile, the Foundation Profile, is available for CDC.

**Profile 1: KJava**

KJava is Sun's proprietary profile and contains the KJava API. The KJava profile is built on top of the CLDC configuration. The KJava virtual machine, KVM, accepts the same byte codes and class file format as the classic J2SE virtual machine. KJava contains a Sun-specific API that runs on the Palm OS. The KJava API has a great deal in common with the J2SE Abstract Windowing Toolkit (AWT). However, because it is not a standard J2ME package, its main package is com.sun.kjava. We'll learn more about the KJava API later in this tutorial when we develop some sample applications.

**Profile 2: MIDP**

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like KJava, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application

development. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

\* java.lang

\* java.io

\* java.util

\* javax.microedition.io

\* javax.microedition.lcdui

\* javax.microedition.midlet

\* javax.microedition.rms

**5. SYSTEM DESIGN & DEVELOPMENT**

Data Flow Diagram / Use Case Diagram / Flow Diagram

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.

Data Flow Diagram

Use Case diagram:



Class Diagram:



Sequence Diagram:



Activity Diagram:



6.SAMPLE CODE

Admin\_login.jsp :

<%@ include file="connect.jsp" %>

<%@ page import="java.util.Date" %>

<%

String username=request.getParameter("adminid");

String Password=request.getParameter("pass");

try

{

String sql="SELECT \* FROM Admin where username='"+username+"' and password='"+Password+"'";

Statement stmt = connection.createStatement();

ResultSet rs =stmt.executeQuery(sql);

String utype="";

if(rs.next())

{

response.sendRedirect("AdminMenu.htm");

}

else

{

response.sendRedirect("wronglogin.html");

}

}

catch(Exception e)

{out.print(e);}

%>

Message.jsp:

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@ include file="connect.jsp" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Insert title here</title>

<style type="text/css">

<!--

.style1 {

color: #FF6600;

font-weight: bold;

font-size: 24px;

}

.style2 {

color: #0000FF;

font-weight: bold;

font-size: 18px;

}

-->

</style>

<script type='text/javascript'>

function saveTextAsFile()

{

var textToWrite = document.getElementById("textarea").value;

var textFileAsBlob = new Blob([textToWrite], {type:'text/plain'});

var fileNameToSaveAs = "Message.txt";

var downloadLink = document.createElement("a");

downloadLink.download = fileNameToSaveAs;

downloadLink.innerHTML = "Download File";

if (window.webkitURL != null)

{

// Chrome allows the link to be clicked

// without actually adding it to the DOM.

downloadLink.href = window.webkitURL.createObjectURL(textFileAsBlob);

}

else

{

// Firefox requires the link to be added to the DOM

// before it can be clicked.

downloadLink.href = window.URL.createObjectURL(textFileAsBlob);

downloadLink.onclick = destroyClickedElement;

downloadLink.style.display = "none";

document.body.appendChild(downloadLink);

}

downloadLink.click();

}

function destroyClickedElement(event)

{

document.body.removeChild(event.target);

}

</script>

</head>

<body>

<%

String s1="",s2="",s3="",s4="",s5="",s6="",s7="",s8,s9="",s10,s11,s12,s13;

int i=0,j=0,k=0;

try

{

int id= Integer.parseInt(request.getParameter("uid"));

String query="select \* from message where id="+id+" ";

Statement st=connection.createStatement();

ResultSet rs=st.executeQuery(query);

while ( rs.next() )

{

s6=rs.getString(6);

s7=rs.getString(7);

%>

<p align="center" class="style1">Message Details </p>

<table width="200" border="1" align="center">

<tr>

<td><span class="style2">Title Name </span></td>

<td><%= s6 %></td>

</tr>

<tr>

<td><span class="style2">Message </span></td>

<td><textarea name="textarea" id="textarea" cols="50" rows="15"><%= s7 %></textarea></td>

</tr>

<tr>

<td>&nbsp;</td>

<td>&nbsp;</td>

</tr>

<tr>

<td>&nbsp;</td>

<td><button onClick="saveTextAsFile()">Download</button></td>

</tr>

</table>

</body>

<%

}

connection.close();

}

catch(Exception e)

{

out.println(e.getMessage());

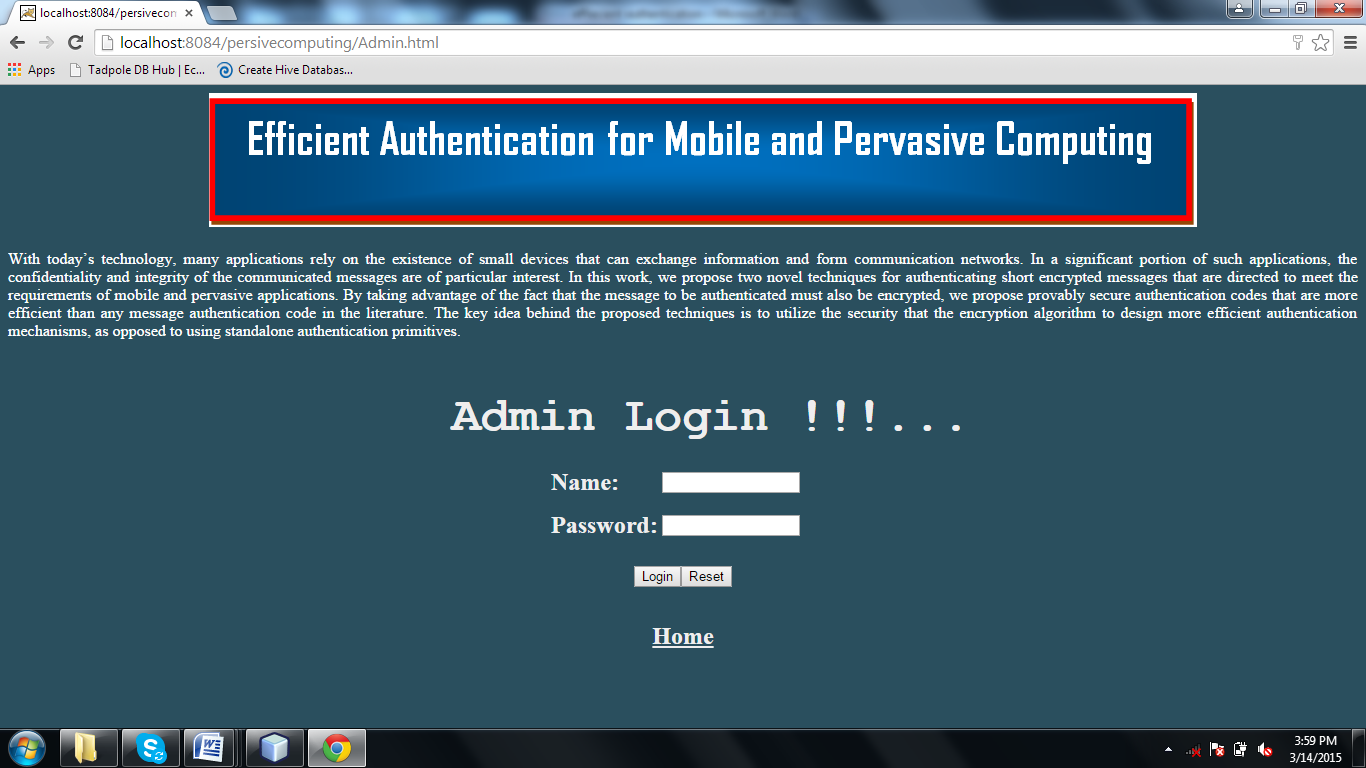
}

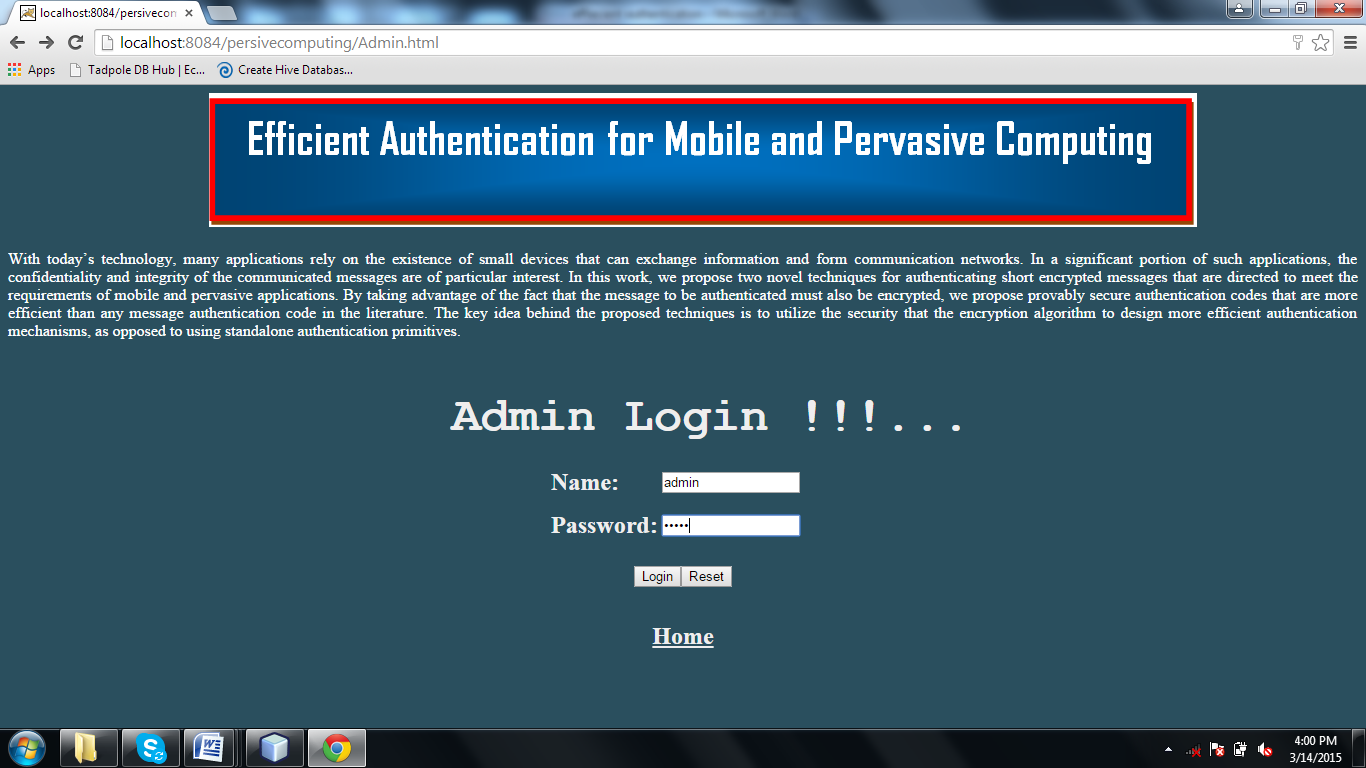
%>

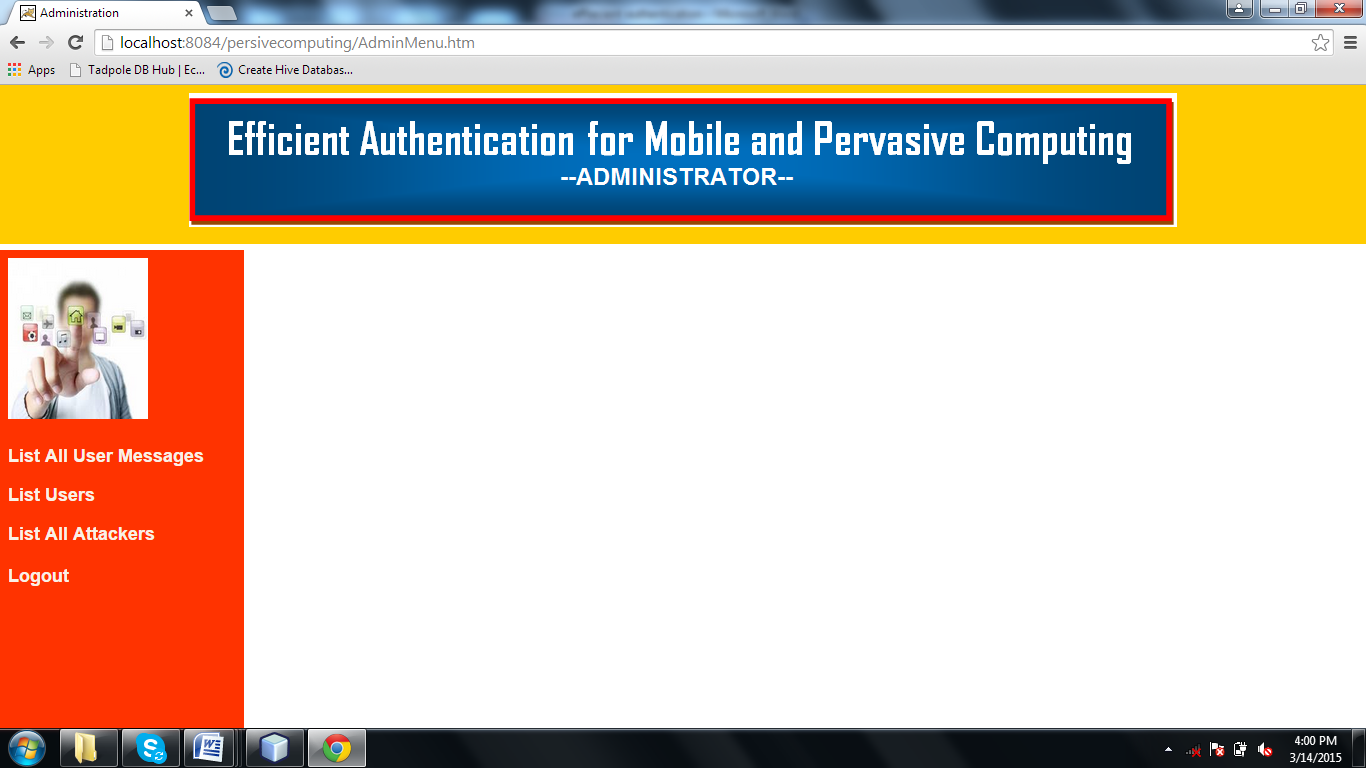
</html>

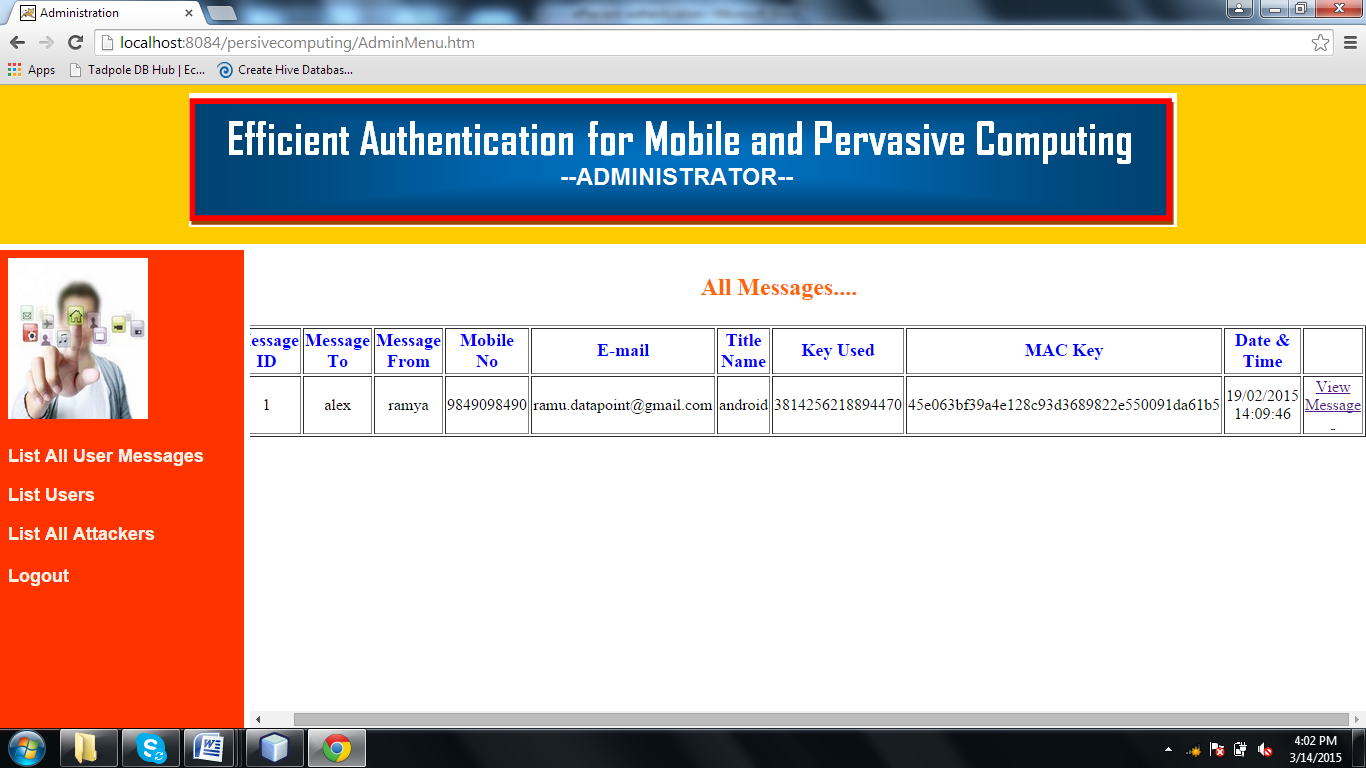
7. SCREEN LAYOUT

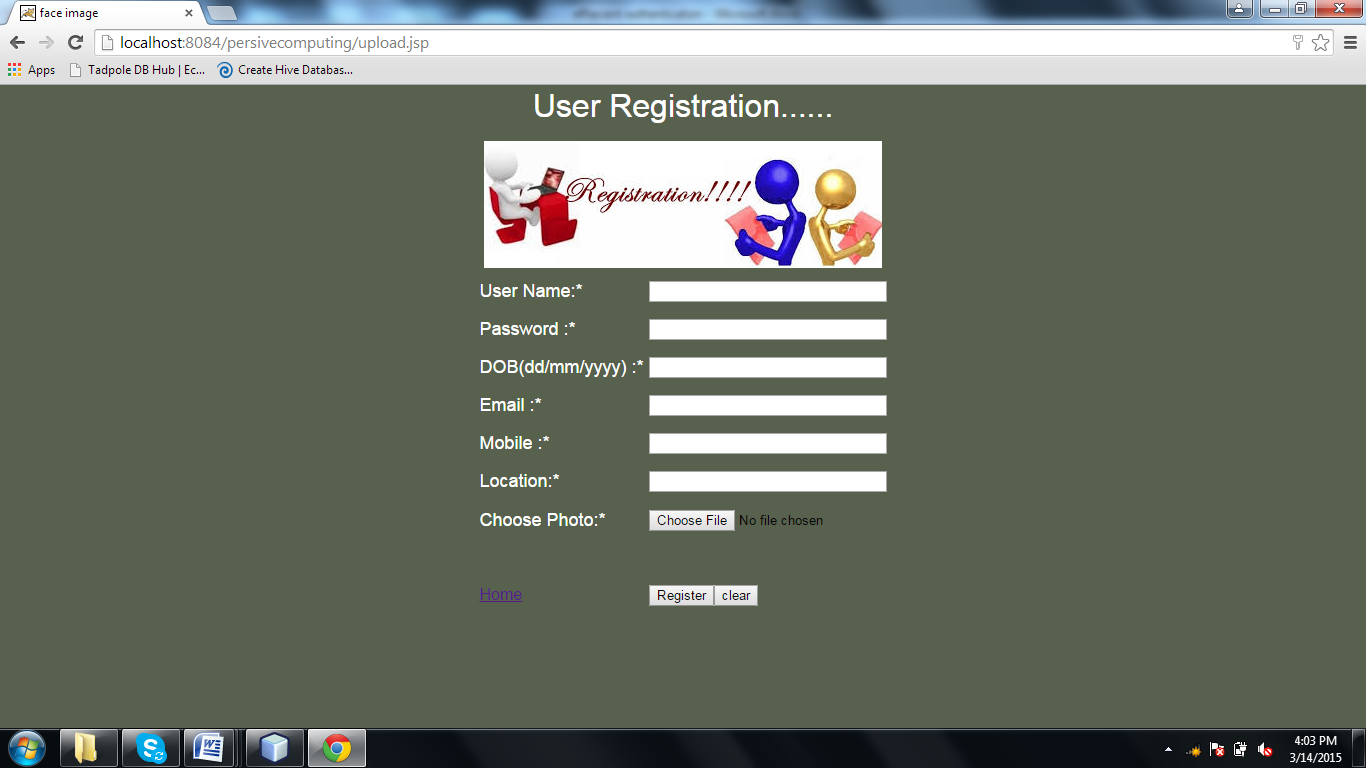


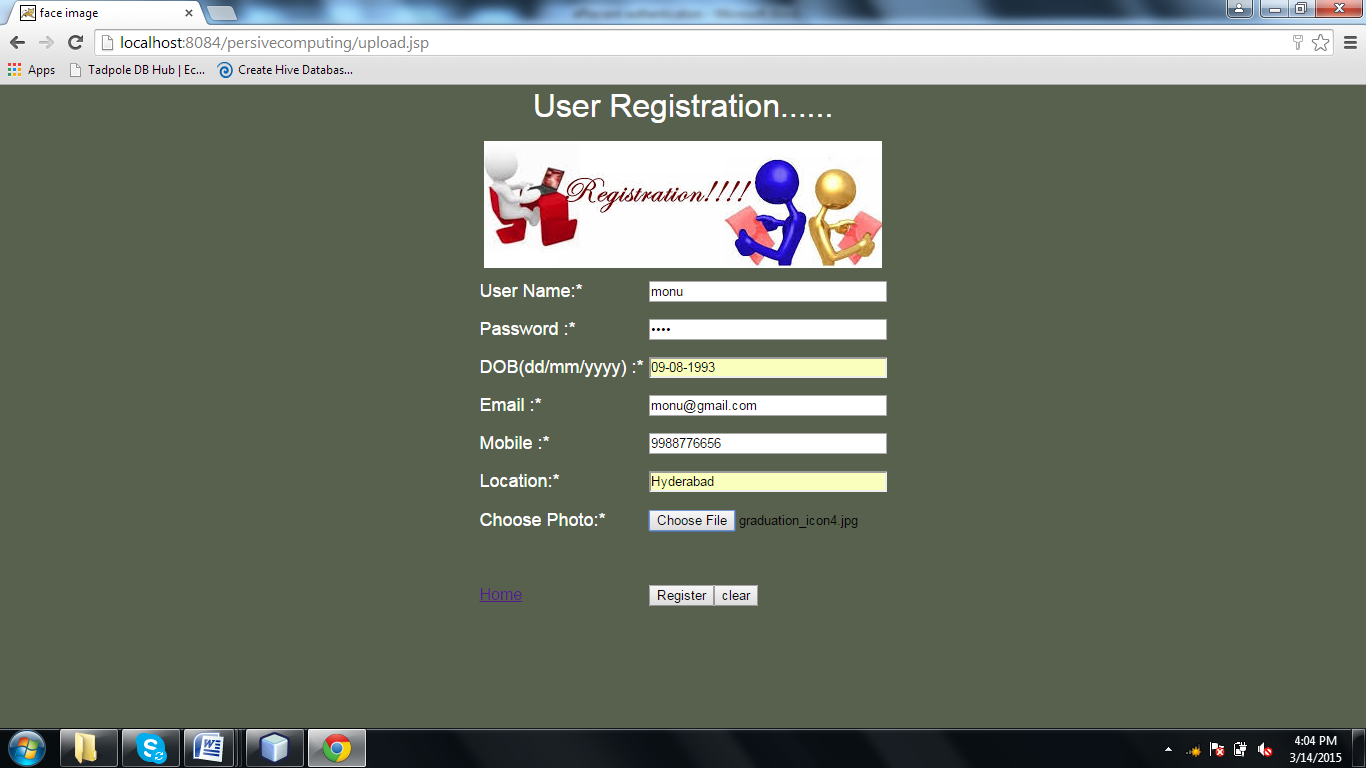


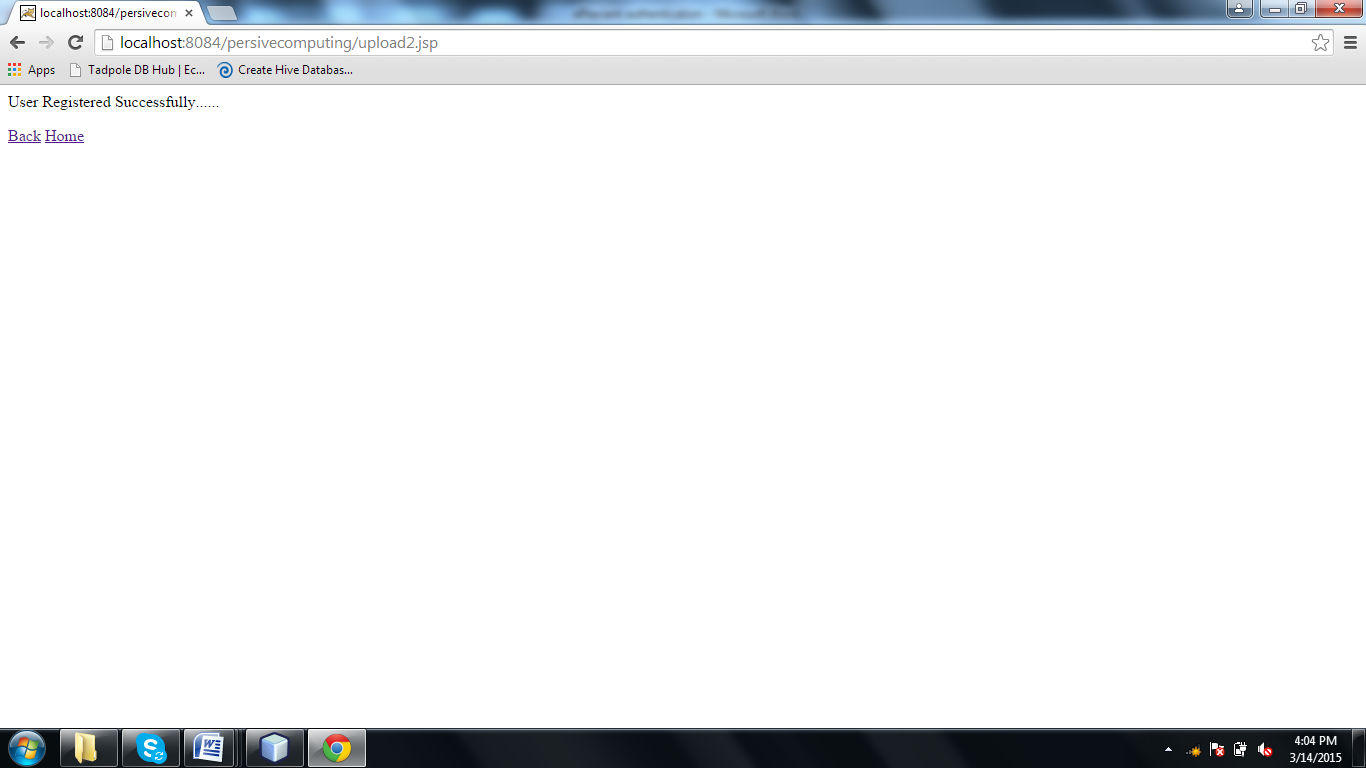


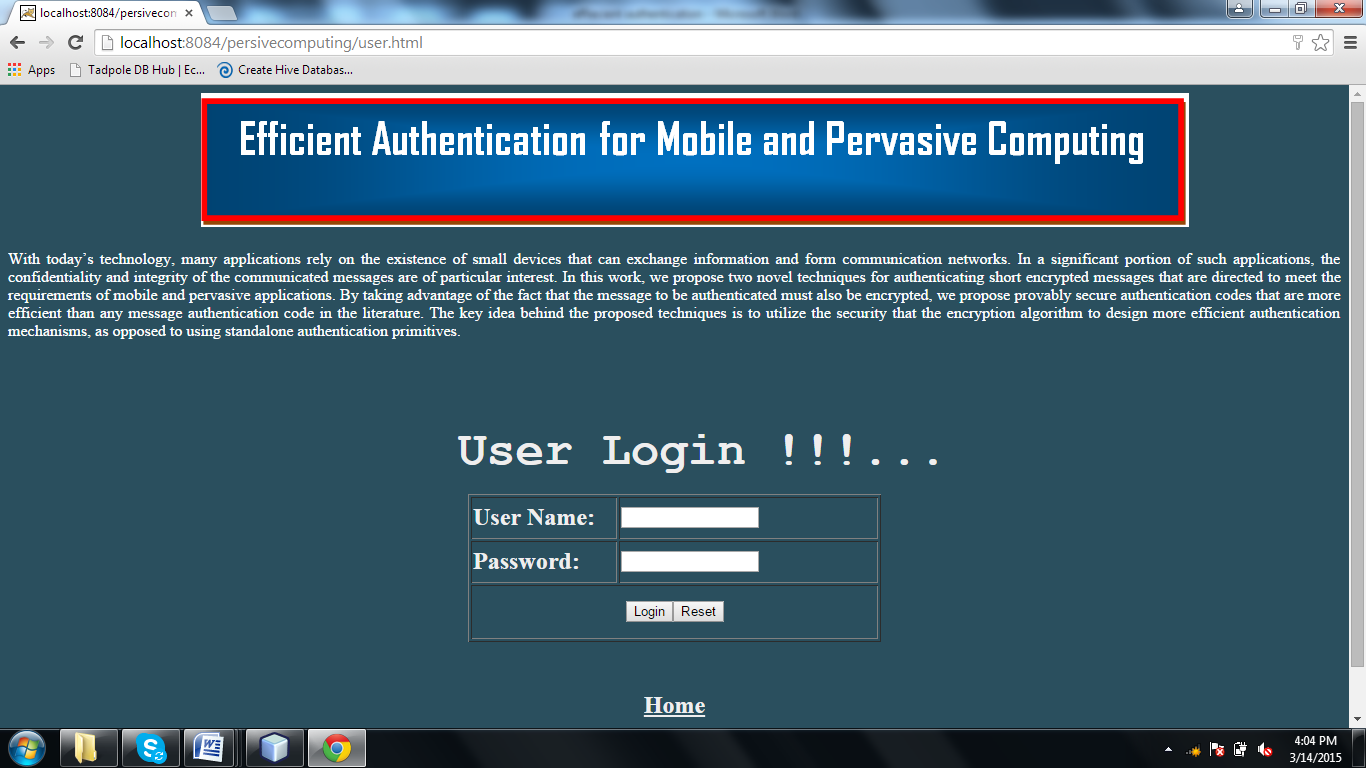


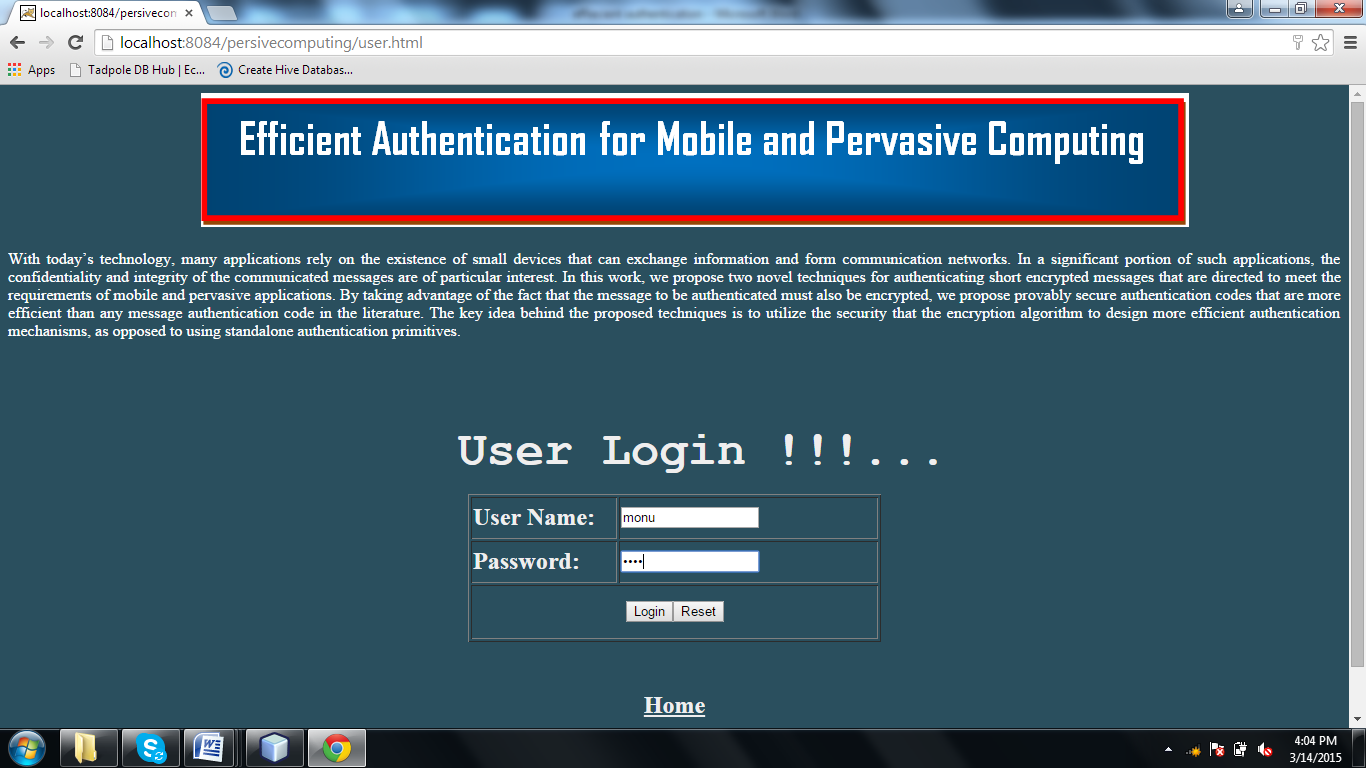


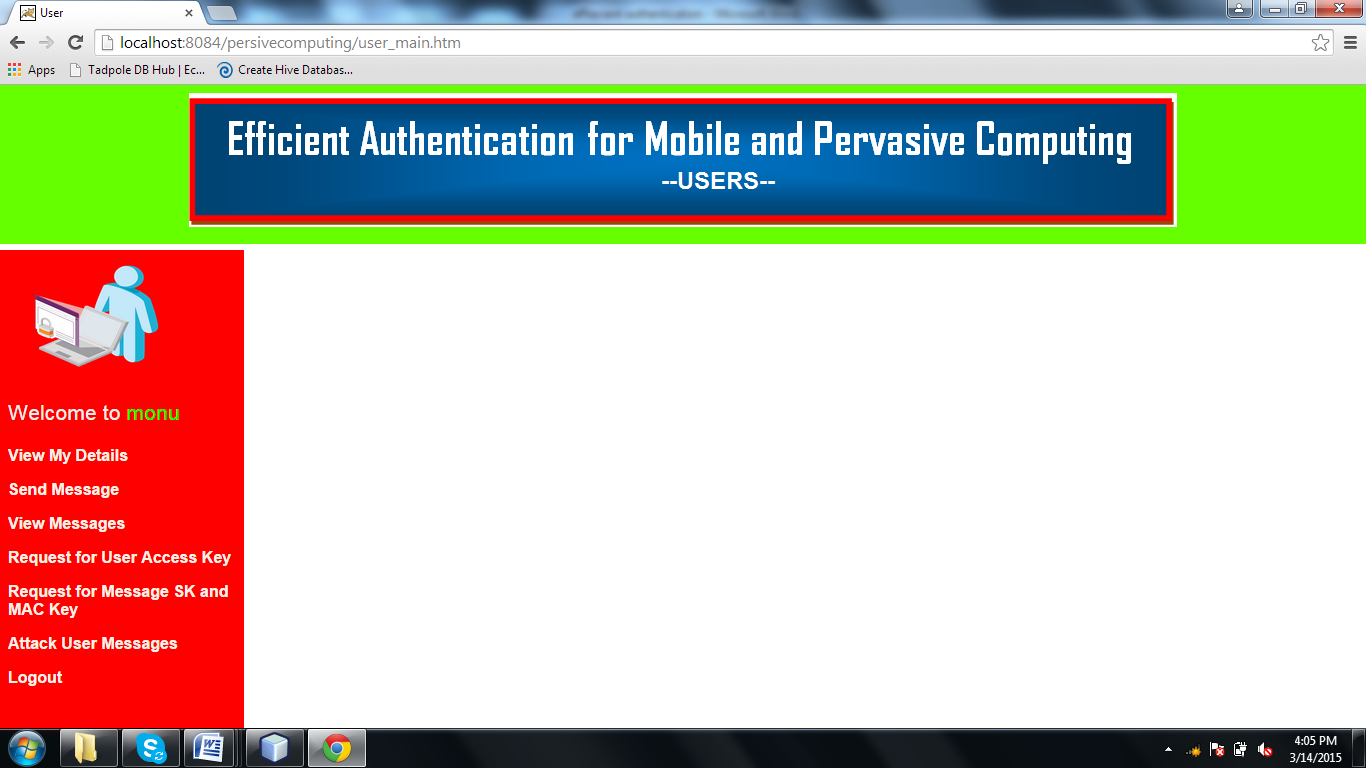


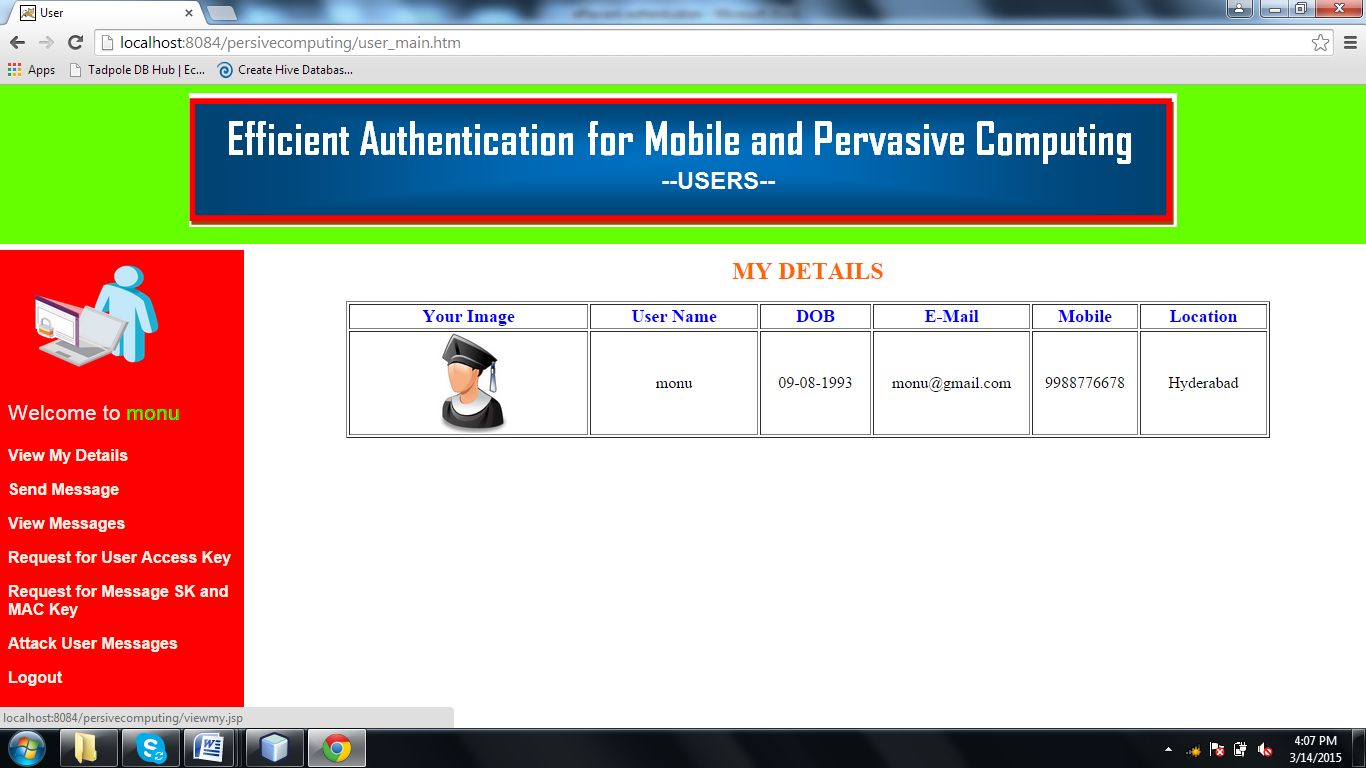


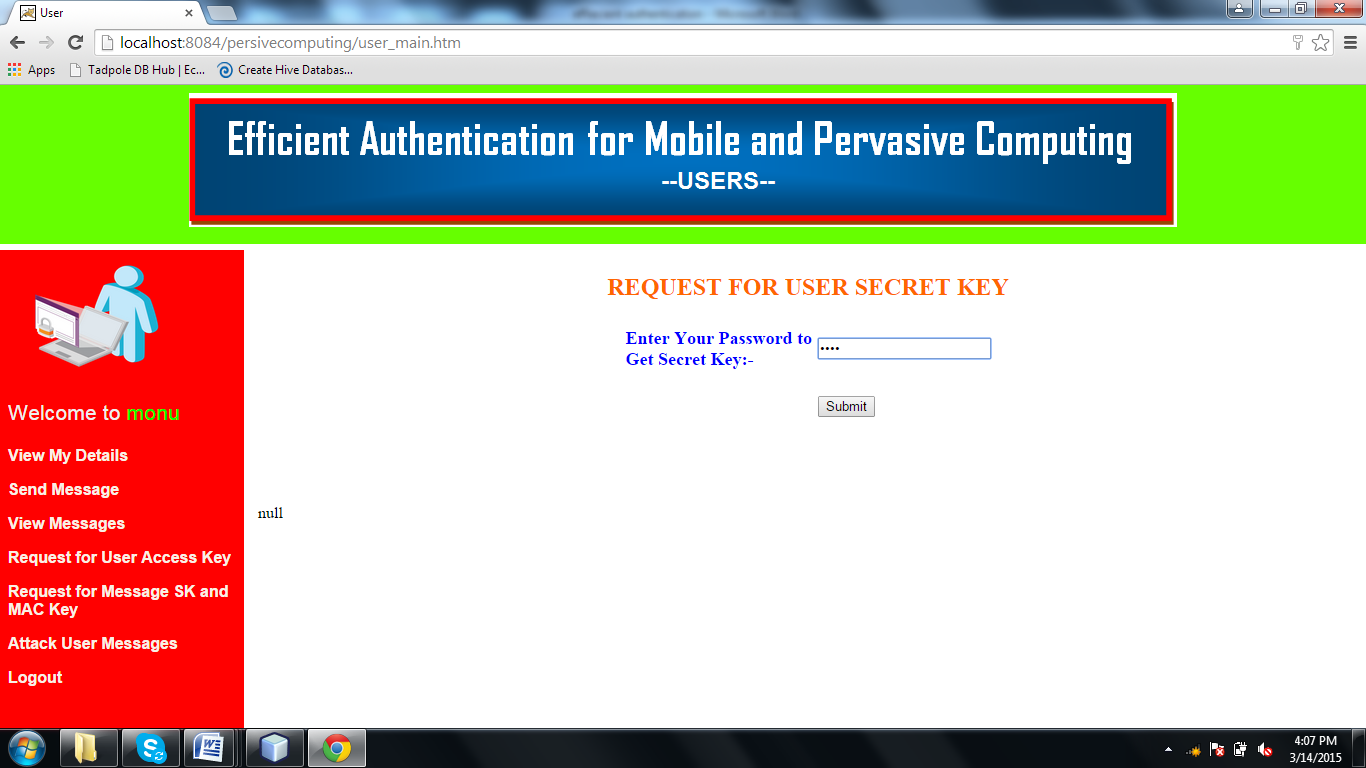


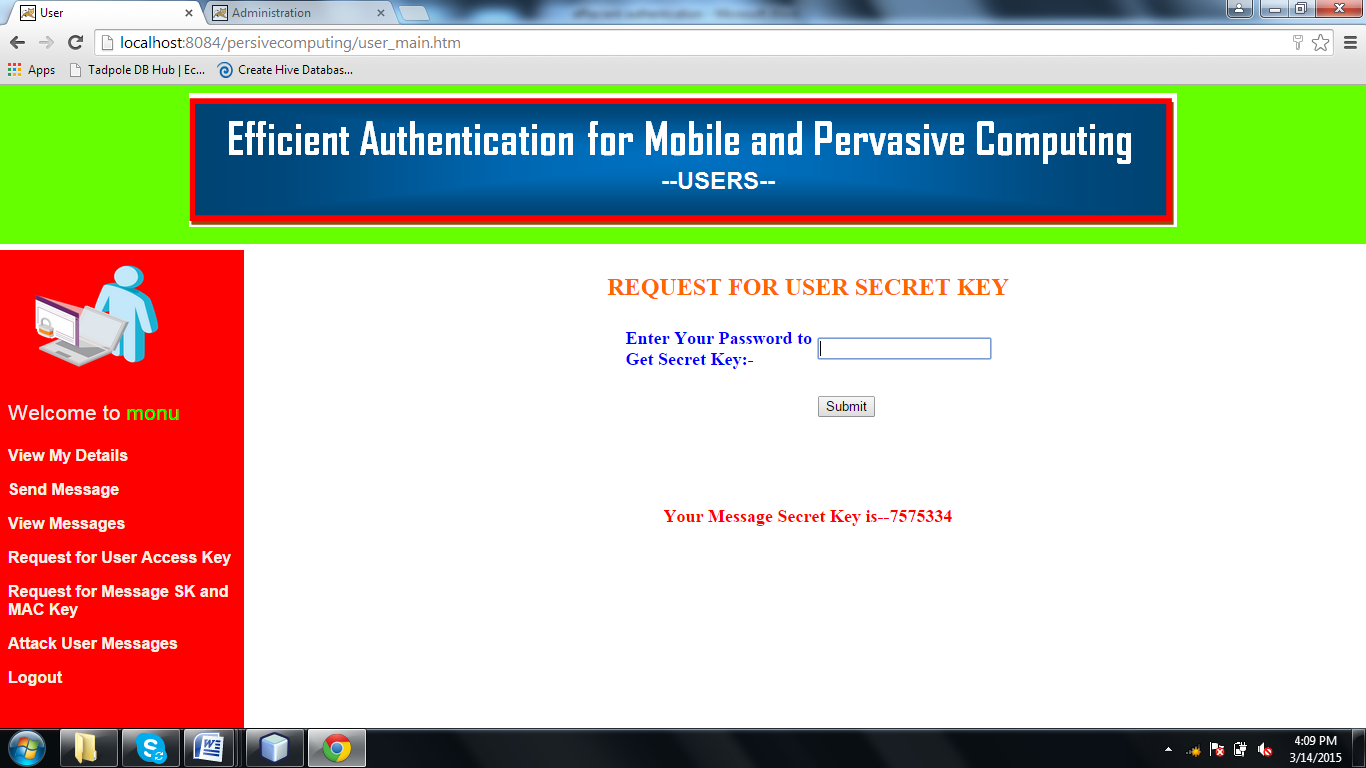


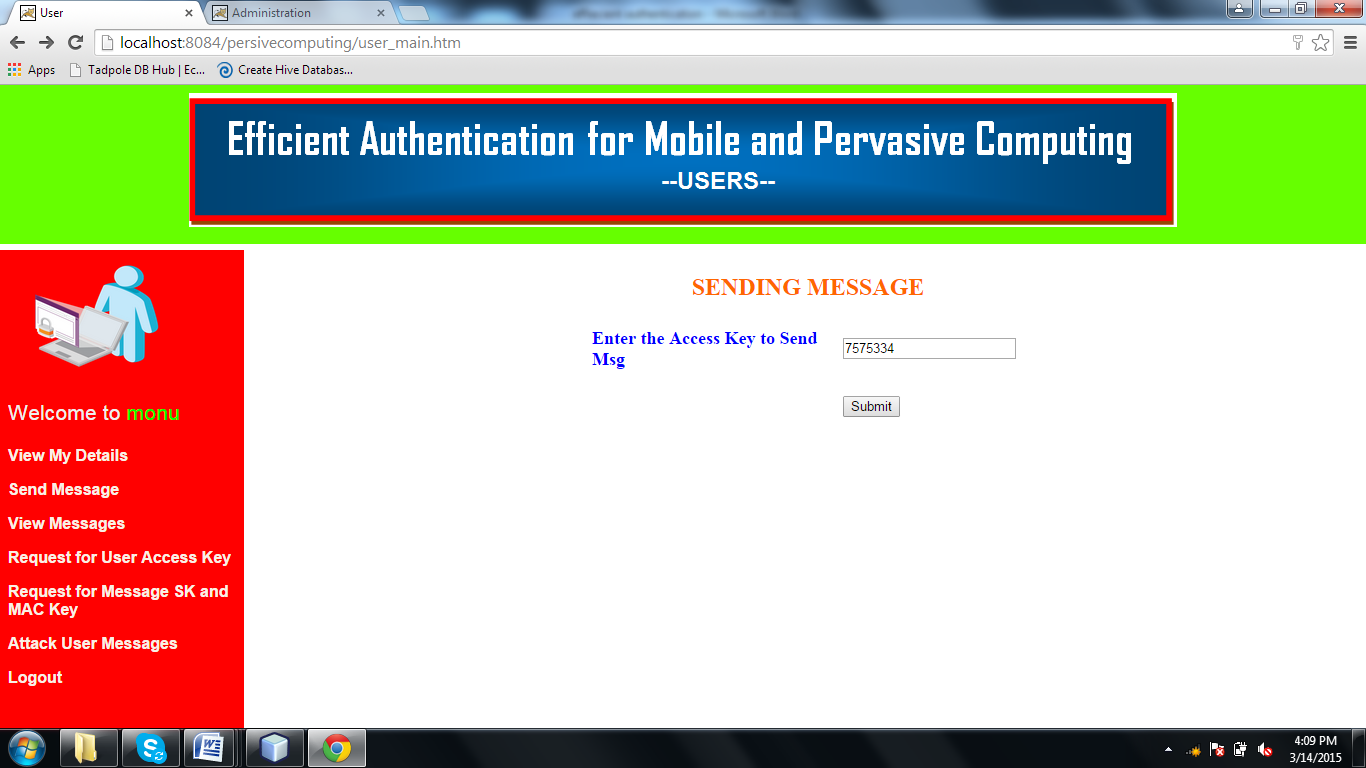












### 6. SYSTEM TESTING

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 satisfaction, 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 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.

**6.1 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.

# 6.2 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.

**6.3 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.

**SYSTEM TESTING**

### TESTING METHODOLOGIES

The following are the Testing Methodologies:

* **Unit Testing.**
* **Integration Testing.**
* **User Acceptance Testing.**
* **Output Testing.**
* **Validation Testing.**

**Unit Testing**

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module’s control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

**Integration Testing**

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

**The following are the types of Integration Testing:**

**1)Top Down Integration**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

**2. Bottom-up Integration**

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

* The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
* A driver (i.e.) the control program for testing is written to coordinate test case input and output.
* The cluster is tested.
* Drivers are removed and clusters are combined moving upward in the program structure

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

**OTHER TESTING METHODOLOGIES**

**User Acceptance Testing**

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

**Output Testing**

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

**Validation Checking**

Validation checks are performed on the following fields.

**Text Field:**

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

**Numeric Field:**

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

**Preparation of Test Data**

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

**Using Live Test Data:**

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

**Using Artificial Test Data:**

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

**USER TRAINING**

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

**MAINTAINENCE**

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user’s requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

**TESTING STRATEGY :**

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

**SYSTEM TESTING:**

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

**UNIT TESTING:**

In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module.

In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this.The future holds a lot to offer to the development and refinement of this project.

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

**OBJECTIVES**

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user

will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

**OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the
* Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

**IMPLEMENTATION**

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it’s constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

**CONCLUSION**

In this work, a new technique for authenticating short encrypted messages is proposed. The fact that the message to be authenticated must also be encrypted is used to deliver a random nonce to the intended receiver via the cipher text. This allowed the design of an authentication code that benefit from the simplicity of unconditionally secure authentication without the need to manage one-time keys. In particular, it has been demonstrated in this paper that authentication tags can be computed with one addition and a one modular multiplication. Given that messages are relatively short, addition and modular multiplication can be performed faster than existing computationally secure MACs in the literature of cryptography. When devices are equipped with block ciphers to encrypt messages,

a second technique that utilizes the fact that block ciphers can be modeled as strong pseudorandom permutations is proposed to authenticate messages using a single modular addition. The proposed schemes are shown to be orders of magnitude faster, and consume orders of magnitude less energy than traditional MAC algorithms. Therefore, they are more suitable to be used in computationally constrained mobile and pervasive devices.

**REFFERENCES**

[1] J. Carter and M. Wegman, “Universal classes of hash functions,” in Proceedings of the ninth annual ACM symposium on Theory of computing–STOC’77. ACM, 1977, pp. 106–112.

[2] M. Wegman and J. Carter, “New classes and applications of hash functions,” in 20th Annual Symposium on Foundations of Computer Science–FOCS’79. IEEE, 1979, pp. 175–182.

[3] L. Carter and M. Wegman, “Universal hash functions,” Journal of Computer and System Sciences, vol. 18, no. 2, pp. 143–154, 1979.

[4] M. Wegman and L. Carter, “New hash functions and their use in authentication and set equality,” Journal of Computer and System Sciences, vol. 22, no. 3, pp. 265–279, 1981.

[5] J. Bierbrauer, “A2-codes from universal hash classes,” in Advances in Cryptology–EUROCRYPT’95, vol. 921, Lecture Notes in Computer Science. Springer, 1995, pp. 311–318.

[6] M. Atici and D. Stinson, “Universal Hashing and Multiple Authentication,” in Advances in Cryptology–CRYPTO’96, vol. 96, Lecture Notes in Computer Science. Springer, 1996, pp. 16–30.

[7] T. Helleseth and T. Johansson, “Universal hash functions from exponential sums over finite fields and Galois rings,” in Advances in cryptology– CRYPTO’96, vol. 1109, Lecture Notes in Computer Science. Springer, 1996, pp. 31–44.

[8] V. Shoup, “On fast and provably secure message authentication based on universal hashing,” in Advances in Cryptology–CRYPTO’96, vol. 1109, Lecture Notes in Computer Science. Springer, 1996, pp. 313–328.

[9] J. Bierbrauer, “Universal hashing and geometric codes,” Designs, Codes and Cryptography, vol. 11, no. 3, pp. 207–221, 1997.

[10] B. Alomair, A. Clark, and R. Poovendran, “The Power of Primes: Security of Authentication Based on a Universal Hash-Function Family,” Journal of Mathematical Cryptology, vol. 4, no. 2, 2010.

[11] B. Alomair and R. Poovendran, “E-MACs: Towards More Secure and More Efficient Constructions of Secure Channels,” in the 13th International Conference on Information Security and Cryptology – ICISC’10. Springer, 2010.

[12] FIPS 113, “Computer Data Authentication,” Federal Information Processing Standards Publication, 113, 1985.

[13] ISO/IEC 9797-1, “Information technology – Security techniques –Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher,” 1999.

[14] M. Dworkin, “Recommendation for block cipher modes of operation: The CMAC mode for authentication,” 2005.

[15] T. Iwata and K. Kurosawa, “omac: One-key cbc mac,” in Fast Software Encryption–FSE’03, vol. 2887, Lecture notes in computer science. Springer, 2003, pp. 129–153.

[16] M. Bellare, R. Guerin, and P. Rogaway, “XOR MACs: New methods for message authentication using finite pseudorandom functions,” in Advances in Cryptology–CRYPTO’95, vol. 963, Lecture Notes in Computer Science. Springer, 1995, pp. 15–28.

[17] P. Rogaway and J. Black, “PMAC: Proposal to NIST for a parallelizable message authentication code,” 2001.

[18] M. Bellare, J. Kilian, and P. Rogaway, “The Security of the Cipher Block Chaining Message Authentication Code,” Journal of Computer and System Sciences, vol. 61, no. 3, pp. 362–399, 2000.

[19] B. Preneel and P. Van Oorschot, “On the security of iterated message authentication codes,” IEEE Transactions on Information theory, vol. 45, no. 1, pp. 188–199, 1999.

[20] P. Rogaway, “Comments on NISTs RMAC Proposal,” 2002.

[21] G. Tsudik, “Message authentication with one-way hash functions,” ACM SIGCOMM Computer Communication Review, vol. 22, no. 5, p. 38, 1992.

[22] M. Bellare, R. Canetti, and H. Krawczyk, “Keying Hash Functions for Message Authentication,” in Advances in Cryptology–CRYPTO’96, vol. 96, Lecture Notes in Computer Science. Springer, 1996, pp. 1–15.

[23] FIPS 198, “The Keyed-Hash Message Authentication Code (HMAC),” Federal Information Processing Standards Publication, vol. 198, 2002.

[24] B. Preneel and P. Van Oorschot, “MDx-MAC and building fast MACs from hash functions,” in Advances in Cryptology-CRYPTO’95, vol. 963, Lecture Notes in Computer Science. Springer, 1995, pp. 1–14.

[25] ISO/IEC 9797-2, “Information technology – Security techniques –Message Authentication Codes (MACs) – Part 2: Mechanisms using a dedicated hash-function,” 2002.

[26] A. Bosselaers, R. Govaerts, and J. Vandewalle, “Fast hashing on the Pentium,” in Advances in Cryptology-CRYPTO’96, vol. 1109, Lecture Notes in Computer Science. Springer, 1996, pp. 298–312.

[27] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway, “UMAC:Fast and Secure Message Authentication,” in Advances in Cryptology–CRYPTO’99, vol. 1666, Lecture Notes in Computer Science. Springer, 1999, pp. 216–233.

[28] D. Bernstein, “The Poly1305-AES message-authentication code,” in Proceedings of Fast Software Encryption–FSE’05, vol. 3557, Lecture Notes in Computer Science. Springer, 2005, pp. 32–49.

[29] D. McGrew and J. Viega, “The security and performance of the Galois/Counter Mode (GCM) of operation,” in Progress in Cryptology- INDOCRYPT’04, vol. 3348, Lecture notes in computer science. Springer, 2004, pp. 343–355.

[30] S. Halevi and H. Krawczyk, “MMH: Software message authentication in the Gbit/second rates,” in Proceedings of Fast Software Encryption– FSE’97, vol. 1267, Lecture notes in computer science. Springer, 1997,pp. 172–189.

[31] M. Etzel, S. Patel, and Z. Ramzan, “Square hash: Fast message authentication

via optimized universal hash functions,” in Advances in Cryptology–CRYPTO’99, vol. 1666, Lecture Notes in Computer Science. Springer, 1999, pp. 234–251.

[32] J. Kaps, K. Yuksel, and B. Sunar, “Energy scalable universal hashing,” IEEE Transactions on Computers, vol. 54, no. 12, pp. 1484–1495, 2005.

[33] D. Bernstein, “Floating-point arithmetic and message authentication,” Available at http:// cr.yp.to/hash127.html, 2004.

[34] H. van Tilborg, Encyclopedia of cryptography and security. Springer, 2005.

[35] T. Krovetz, “http://fastcrypto.org/umac/,” 2006.

[36] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” Computer networks, vol. 38, no. 4, pp. 393–422, 2002.

[37] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler, “SPINS: Security protocols for sensor networks,” Wireless networks, vol. 8, no. 5, pp. 521–534, 2002.

[38] A. Perrig, J. Stankovic, and D. Wagner, “Security in wireless s Sensor networks,” Communications of the ACM, vol. 47, no. 6, pp. 53–57, 2004.

[39] EPCglobal Inc., “Class-1 generation-2 uhf radio frequency identification protocol standard version 1.2.0,” 2008.

[40] S. Sarma, S. Weis, and D. Engels, “RFID systems and security and privacy implications,” Cryptographic Hardware and Embedded Systems-CHES 2002, pp. 1–19, 2003.

[41] A. Juels, “RFID security and privacy: A research survey,” IEEE Journal on Selected Areas in Communications, vol. 24, no. 2, pp. 381–394, 2006.

[42] P. Peris-Lopez, J. Hernandez-Castro, J. Estevez-Tapiador, and A. Ribagorda, “RFID systems: A survey on security threats and proposed solutions,” in Personal Wireless Communications. Springer, 2006, pp. 159–170.

[43] Y. Zhang, “A design proposal of security architecture for medical body sensor networks,” in Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on. IEEE, 2006, pp. 4–90.

[44] K. Venkatasubramanian, A. Banerjee, and S. Gupta, “Ekg-based key agreement in body sensor networks,” in INFOCOM Workshops 2008, IEEE. IEEE, 2008, pp. 1–6.

[45] C. Tan, H. Wang, S. Zhong, and Q. Li, “Body sensor network security: an identity-based cryptography approach,” in Proceedings of the first ACM conference on Wireless network security. ACM, 2008, pp. 148–153.

[46] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, “Strong Authentication for RFID Systems using the AES Algorithm,” in Cryptographic Hardware and Embedded Systems–CHES’04, vol. 3156, Lecture Notes in Computer Science. Springer, 2004, pp. 357–370.

[47] C. H. Lim and T. Korkishko, “mCrypton - A Lightweight Block Cipher For Security of Low-Cost RFID Tags and Sensors,” in Workshop on Information Security Applications–WISA’05, ser. Lecture Notes in Computer Science, vol. 3786. Springer, 2005, pp. 243–258.

[48] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee, “HIGHT: A New Block Cipher Suitable for Low-Resource Device,” in Cryptographic Hardware and Embedded Systems–CHES’06, ser. Lecture Notes in Computer Science, vol. 4249. Springer, 2006, pp. 46–59.

[49] A. Poschmann, G. Leander, K. Schramm, and C. Paar, “A Family of Light-Weight Block Ciphers Based on DES Suited for RFID Applications,” in Workshop on RFID Security–RFIDSec’06. Ecrypt, 2006.

[50] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw,

Y. Seurin, and C. Vikkelsoe, “PRESENT: An Ultra-Lightweight Block Cipher,” in Cryptographic Hardware and Embedded Systems– CHES’07, vol. 4727, Lecture Notes in Computer Science. Springer, 2007, pp. 450–466.

[51] F. Mac´e, F.-X. Standaert, and J.-J. Quisquater, “ASIC Implementations

of the Block Cipher SEA for Constrained Applications,” in Workshop on RFID Security–RFIDSec’07, 2007.

[52] M. O’Neill (McLoone), “Low-Cost SHA-1 Hash Function Architecture for RFID Tags,” in Workshop on RFID Security–RFIDSec’08, 2008.

[53] A. Shamir, “SQUASH–A New MAC with Provable Security Properties for Highly Constrained Devices Such as RFID Tags,” in Fast Software Encryption–FSE’08, vol. 5086, Lecture Notes in Computer Science. Springer, 2008, pp. 144–157.

[54] A. Bogdanov, G. Leander, C. Paar, A. Poschmann, M. Robshaw, and Y. Seurin, “Hash Functions and RFID Tags : Mind The Gap,” in Proceedings of the 10th International Workshop Cryptographic Hardware and Embedded Systems–CHES’08, ser. Lecture Notes in Computer Science, vol. 5154. Springer, 2008, pp. 283–299.

[55] E. B. Kavun and T. Yalcin, “A Lightweight Implementation of Keccak Hash Function for Radio-Frequency Identification Applications,” in Workshop on RFID Security–RFIDSec’10, 2010.

[56] O. Goldreich, Foundations of Cryptography. Cambridge University Press, 2001.

[57] S. Goldwasser and S. Micali, “Probabilistic encryption,” Journal of Computer and System Sciences, vol. 28, no. 2, pp. 270–299, 1984.

[58] T. Kohno, J. Viega, and D. Whiting, “CWC: A high-performance conventional authenticated encryption mode,” in Fast Software Encryption– FSE’04, vol. 3017, Lecture Notes in Computer Science. Springer, 2004, pp. 408–426.

[59] M. Luby and C. Rackoff, “How to Construct Pseudorandom Permutations

from Pseudorandom Functions,” SIAM Journal on Computing, vol. 17, pp. 373–386, 1988.

[60] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, “A concrete security treatment of symmetric encryption,” in 38th Annual Symposium on Foundation of Computer Science–FOCS’97. IEEE Computer Society, 1997, pp. 394–403.

[61] J. Daemen and V. Rijmen, The design of Rijndael: AES–the Advanced

Encryption Standard. Springer Verlag, 2002.

[62] S. Schwarz, “The role of semigroups in the elementary theory of numbers,” Math. Slovaca, vol. 31, no. 4, pp. 369–395, 1981.

[63] Z. Liu and D. Peng, “True Random Number Generator in RFID Systems Against Traceability,” in IEEE Consumer Communications and Networking Conference–CCNS,06, vol. 1. IEEE, 2006, pp. 620–624.

[64] D. Holcom,W. Burleson, and K. Fu, “Initial SRAM state as a Fingerprint and Source of True Random Numbers for RFID Tags,” in Workshop on RFID Security–RFIDSec’07, 2007.

[65] D. Holcomb, W. Burleson, and K. Fu, “Power-up SRAM State as an Identifying Fingerprint and Source of True Random Numbers,” IEEE Transactions on Computers, vol. 58, no. 9, 2009.

[66] C. Petrie and J. Connelly, “A noise-based IC random number generator for applications in cryptography,” IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 47, no. 5, pp. 615–621, 2000.

[67] S. Callegari, R. Rovatti, and G. Setti, “Embeddable ADC-based true random number generator for cryptographic applications exploiting nonlinear signal processing and chaos,” IEEE Transactions on Signal Processing, vol. 53, no. 2 Part 2, pp. 793–805, 2005.

[68] A. Francillon, C. Castelluccia, and P. Inria, “TinyRNG: A cryptographic random number generator for wireless sensors network nodes,” in Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks– WiOpt’07. Citeseer, 2007, pp. 1–7.

[69] J. Nakajima and M. Matsui, “Performance analysis and parallel implementation of dedicated hash functions,” in Advances in Cryptology– EUROCRYPT 2002. Springer, 2002, pp. 165–180.

[70] B. Preneel, “Using Cryptography Well,” Printed handout available at http://secappdev.org/handouts/2010/Bart%20Preneel/using crypto well

pdf, 2010.

[71] J. Großsch¨adl, R. Avanzi, E. Savas¸, and S. Tillich, “Energy-efficient software implementation of long integer modular arithmetic,” in Proceedings of the 7th international conference on Cryptographic hardware and embedded systems – CHES’05, vol. 3659. Springer-Verlag, 2005, pp. 75–90.

[72] N. Ferguson, D. Whiting, B. Schneier, J. Kelsey, and T. Kohno, “Helix: Fast encryption and authentication in a single cryptographic primitive,” in Proceedings of Fast Software Encryption–FSE’03, vol. 2887, Lecture notes in computer science. Springer, 2003, pp. 330–346.

[73] D. Whiting, B. Schneier, S. Lucks, and F. Muller, “Phelix-fast encryption and authentication in a single cryptographic primitive, eSTREAM,” ECRYPT Stream Cipher Project, Report 2005/020, [www.ecrypt.eu.org/](http://www.ecrypt.eu.org/) stream, 2005.

[74] F. Muller, “Differential attacks against the Helix stream cipher,” in Fast Software Encryption–FSE’04, vol. 3017, Lecture Notes in Computer Science. Springer, 2004, pp. 94–108.

[75] S. Paul and B. Preneel, “Solving systems of differential equations of addition,” in Australasian Conference on Information Security and Privacy–ICISP’05, vol. 3574, Lecture Notes in Computer Science. Springer, 2005, pp. 75–88.

[76] ——, “Near Optimal Algorithms for Solving Differential Equations of Addition with Batch Queries,” in Progress in Cryptology– INDOCRYPT’05, vol. 3797, Lecture Notes in Computer Science. Springer, 2005, pp. 90–103.

[77] H. Wu and B. Preneel, “Differential-linear attacks against the stream cipher Phelix,” in Fast Software Encryption–FSE’07, vol. 4593, Lecture Notes in Computer Science. Springer, 2007, pp. 87–100.

[78] D. Stinson, Cryptography: Theory and Practice. CRC Press, 2006.

[79] M. Bellare and C. Namprempre, “Authenticated Encryption: Relations Among Notions and Analysis of the Generic Composition Paradigm,” Journal of Cryptology, vol. 21, no. 4, pp. 469–491, 2008.

[80] J. Katz and Y. Lindell, Introduction to modern cryptography. Chapman

& Hall/CRC, 2008.

[81] M. F¨urer, “Faster integer multiplication,” in ACM symposium on Theory of computing–STOC’07. ACM, 2007, p. 66.

[82] C. Jutla, “Encryption modes with almost free message integrity,” Journal

of Cryptology, vol. 21, no. 4, pp. 547–578, 2008.

[83] P. Rogaway, M. Bellare, and J. Black, “OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption,” ACM Transactions on Information and System Security, vol. 6, no. 3, pp. 365–403, 2003.

[84] A. Menezes, P. Van Oorschot, and S. Vanstone, Handbook of applied cryptography. CRC, 1997.

[85] B. Alomair and R. Poovendran, “Efficient Authentication for Mobile and Pervasive Computing,” in The 12th International Conference on Information and Communications Security–ICICS’10. Springer, 2010.