A Major Project Report

On

Blockchain-Based Secure Key Management for Mobile Edge Computing

A report submitted in partial fulfillment of the requirements for the award

of the degree of B. Tech

By

Pallikonda Rithvik

(20EG105635)

Gunda Mani Akshaya

(20EG105649)

Kanagala Sai Lokesh

(20EG105705)

Under the guidance of

Mrs. K Rashmi

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ANURAG UNIVERSITY
VENKATAPUR - 500088
TELANGANA
Year: 2023-2024

i

DECLARATION

We hereby declare that the Report entitled **"Blockchain-Based Secure Key Management for Mobile Edge Computing"** submitted for the award of Bachelor of technology Degree is our original work and the Report has not formed the basis for the award of any degree, diploma, associate ship or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma.

Place: Anurag University, Hyderabad

Date:

P. Rithvik (20EG105635)

G. Mani Akshaya (20EG105649)

K. Sai Lokesh (20EG105705)

CERTIFICATE

This is to certify that the Report entitled **"Blockchain-Based Secure Key Management for Mobile Edge Computing"** that is being submitted by P. Rithvik (20EG105635), G. Mani Akshaya (20EG105649), K. Sai Lokesh (20EG105705) in partial fulfillment for the award of B.Tech. in Computer Science and Engineering to the Anurag University is a record of bonafide work carried out by them under my guidance and supervision.

The results embodied in this Report have not been submitted to any other University or Institute for theaward of any degree or diploma.

Signature of Supervisor   **Dean, CSE**

Mrs. K Rashmi

Assistant Professor, CSE

External Examiner

<u>ACKNOWLEDGMENT</u>

ABSTRACT

Mobile edge computing (MEC) is a promising edge technology to provide high bandwidth and low latency shared services and resources to mobile users. However, the MEC infrastructure raises major security concerns when the shared resources involve sensitive and private data of users. This paper proposes a novel blockchain-based key management scheme for MEC that is essential for ensuring secure group communication among the mobile devices as they dynamically move from one subnetwork to another. In the proposed scheme, when a mobile device joins a subnetwork, it first generates lightweight key pairs for digital signature and communication, and broadcasts its public key to neighbouring peer users in the subnetwork blockchain. The blockchain miner in the subnetwork packs all the public key of mobile devices into a block that will be sent to other users in the subnetwork. This enables the system device to communicate with its peers in the subnetwork by encrypting the data with the public key stored in the blockchain. When the mobile device moves to another subnetwork in the tree network, all the devices of the new subnetwork can quickly verify its identity by checking its record in the local or higher hierarchy subnetwork blockchain. Furthermore, when the device leaves the subnetwork, it does not need to do anything and its records will remain in the blockchain which is an append-only database. Theoretical security analysis shows that the proposed scheme can defend against the 51 percent attack and malicious entities in the blockchain network utilizing Proof-of-Work consensus mechanism. Moreover, the backward and forward secrecy is also preserved. Experimental results demonstrate that the proposed scheme outperforms two baselines in terms of computation, communication, and storage.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Mobile edge computing (MEC) is now recognized as a key edge technology for supporting low latency and high bandwidth mobile services with shared resources. Many of these services will involve transferring private and sensitive information of the users among a group of communicating devices, which give rise to major security concerns. Roman et al. and Yi et al. have shown the existing vulnerability and threats in MEC. It is worthwhile pointing out that all participants are assumed to be untrusted in the MEC network. Key management is an essential component for providing security in any network. In a MEC infrastructure, the security of key management should allow nodes to establish pairwise keys in order to facilitate secure communication between the nodes in the network. The mobility of devices brings a new challenge for ensuring group communication in a MEC network, i.e., how to transfer keys to members moving from one subnetwork (group) to another while still remaining in communication.

Since the members of the new subnetwork do not recognize the newly joined member, an additional overhead will be incurred for key generation, key distribution and key storage, i.e., computation overhead, communication overhead and storage overhead, respectively. Meanwhile, the group key management schemes must not only handle dynamic group membership (joining or leaving), but they must also handle dynamic member location (moving). Therefore, there is an urgent requirement for a group key management scheme to protect the security of keys, both for the members joining or leaving the group and for moving across network areas. Key management in MEC has attracted the interest of many researchers in recent years. A typical method in centralized key management schemes is the logical key hierarchy-based scheme that was proposed by several research groups almost at the same time. The key tree adopted in such schemes is able to reduce the number of messages required for key updating. For distributed key management schemes, a typical scheme is the distributed group management scheme that is proposed in. One of the typical schemes in the decentralized key management schemes is the mobile key management scheme for dynamic secure group communication which is proposed in. There exist many challenges in achieving efficient and secure key management in the wireless

environment. The first challenge is dealing with large-scale mobile devices in a wireless network. Secondly, with the increasing mobility of users, the real-time services are demanding higher performance on rekeying process such as recalculating, redistributing and restoring new keys for backward and forward secrecy. Thirdly, the security issues are a major concern especially when a centralized entity is relied upon for ensuring security of the key management system. The emerging blockchain technique offers the potential to build a trusted, fair-minded and decentralized environment for key management scheme in wireless mobile edge networks. Besides, keeping private keys locally and storing public keys in the blockchain can reduce the overhead of rekeying process. Because when users move into a new group, their records (i.e., public keys) can be traced in the previous blocks in local or higher hierarchical blockchain, so that their identities can be verified quickly. Meanwhile, only if the quick verification is failed, the users need to generate a new key pair (i.e., rekeying) which will be appended to the blockchain.

Moreover, the key pair of users will be invalid once they generate a new one, which can reduce the overhead of backward and forward secrecy in the rekeying process. However, blockchain cannot be directly applied to the existing key management schemes, especially in wireless mobile environments, due to the following challenges. Firstly, it is inefficient for the blockchain to store a large volume of data in the blockchain network. Secondly, it is a challenge to adapt the centralized functions of the third party to the decentralized blockchain network. Thirdly, it is a challenge to combine blockchain with a key management scheme in the wireless mobile environment. This paper aims to reduce the rekeying overhead and improve the security of key management in MEC network. In order to deal with all the challenges mentioned above, we analyse the characteristics of key management schemes and blockchain technique, and then we construct a blockchain based key management scheme. In the proposed scheme, we first partition the wireless base stations into groups (i.e., subnetworks), then build a tree network by treating these groups as nodes, according to the number of members (i.e., users) and the resources of the groups. Then in each group, all members maintain a blockchain network. And the data will be collected into the blockchain of the higher hierarchical nodes, and to be gathered into the root node. Therefore, the members of a group can quickly verify the new group member by tracing the records in the local or higher hierarchical blockchain.

1.1 LITERATURE SURVEY

[1] "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges,"

For various reasons, the cloud computing paradigm is unable to meet certain requirements (e.g. low latency and jitter, context awareness, mobility support) that are crucial for several applications (e.g. vehicular networks, augmented reality). To fulfil these requirements, various paradigms, such as fog computing, mobile edge computing, and mobile cloud computing, have emerged in recent years. While these edge paradigms share several features, most of the existing research is compartmentalized; no synergies have been explored. This is especially true in the field of security, where most analyses focus only on one edge paradigm, while ignoring the others. The main goal of this study is to holistically analyses the security threats, challenges, and mechanisms inherent in all edge paradigms, while highlighting potential synergies and venues of collaboration. In our results, we will show that all edge paradigms should consider the advances in other paradigms.

[2] A survey of key management schemes in multi-phase wireless sensor networks," Computer Networks, vol. 105, pp. 60–74, 2016.

Wireless Sensor Networks (WSNs) are the enabling technology for smart cities, intelligent cars and transportation systems, precision agriculture, animal tracking, and all data collection and sensing-based applications. In most WSN applications, new sensor nodes are added to the network by post-deployment to assure network connectivity, to replace dead sensor nodes or to cover more regions in the area of interest. This type of network is called Multi Phase WSNs (MPWSNs). Similarly to classical WSNs, multi-phase WSNs require security mechanisms to ensure their deployment. However, these networks need specific solutions adapted to the multiple deployments of nodes. In this paper, we review, classify and compare the existing key management schemes proposed for this type of sensor network. We illustrate both advantages and disadvantages of each multi-phase key management scheme. Finally, we give some directions to design lightweight robust key management for MPWSNs.

[3] "Dynamic outsourced auditing services for cloud storage based on batch-leaves-authenticated merkle hash tree," IEEE Transactions on Services Computing, vol. 13, no. 3, pp. 451–463, 2017.

Wireless sensor networks include a large number of sensor nodes, which perform the receiving, sending, and processing of the data. Because of the wireless nature of communications in this kind of networks, they are more vulnerable than the wired networks. Therefore, security in wireless sensor networks is vital and requires special instruments and techniques. Hence, the use of security mechanisms that are capable of optimally enhance the security and the lifetime of the network, using the limited capacity of the nodes is absolutely vital. One way of maintaining security in the wireless sensor networks is through key management, which is subdivided into dynamic key management and static key management. In this paper, we review dynamic key management systems in wireless sensor networks and introduce some evaluation criteria in key management systems. Also, we categorize dynamic key management schemes based on the type of keys, key distribution mechanisms, key cryptography methods and network models. The proposed categorization in this paper helps researchers achieve a comprehensive view of dynamic key management schemes in wireless sensor networks. Familiarity with new techniques and different dynamic key management methods allow further research in this area.

[4] "Quantum resource estimates for computing elliptic curve discrete logarithms," in International Conference on the Theory and Application of Cryptology and Information Security. Springer, 2017, pp. 241–270.

We give precise quantum resource estimates for Shor's algorithm to compute discrete logarithms on elliptic curves over prime fields. The estimates are derived from a simulation of a Toffoli gate network for controlled elliptic curve point addition, implemented within the framework of the quantum computing software tool suite LIQ. We determine circuit implementations for reversible modular arithmetic, including modular addition, multiplication and inversion, as well as reversible elliptic curve point addition. We conclude that elliptic curve discrete logarithms on an elliptic curve defined over an -bit prime field can be computed on a quantum computer with at most qubits using a quantum circuit of at most Toffoli gates. We are able to classically simulate the Toffoli networks corresponding to the controlled elliptic curve point addition as the core piece of Shor's algorithm for the NIST standard curves P-192, P-

224, P-256, P-384 and P-521. Our approach allows gate-level comparisons to recent resource estimates for Shor's factoring algorithm. The results also support estimates given earlier by Proos and Zalka and indicate that, for current parameters at comparable classical security levels, the number of qubits required to tackle elliptic curves is less than for attacking RSA, suggesting that indeed ECC is an easier target than RSA.

[5] "Use of elliptic curves in cryptography," in Conference on the theory and application of cryptographic techniques. Springer, 1985, pp. 417–426.

We discuss the use of elliptic curves in cryptography. In particular, we propose an analogue of the Diffie-Hellmann key exchange protocol which appears to be immune from attacks of the style of Western, Miller, and Adleman. With the current bounds for infeasible attack, it appears to be about 20% faster than the Diffie-Hellmann scheme over GF(p). As computational power grows, this disparity should get rapidly bigger.

1.2 IMPLEMENTATION

MODULES:

Owner Module:

Owner will Register with application and Generate Security Key Which Will Be Useful For Users To Join In Group And Login every user will have two keys private and public key show user profile details with keys public and private show block chain key for user after joining in group same group users will have same block chain each user in that group will have same private and public key where anyone who shares data in that group will store in database with same block chain, encryption data with public key and users details with group name if user changes group request will be sent to block chain manager who will assign new block chain key by getting his previous details from group1 request data in group user private key and decrypt and download\ if user changes group he can't download other groups data.

Block Chain Manager:

Block chain manager will login generate block chain for each group view block chain details of each group view users requests for changing subnetwork assign new block chain key.

Node Server:

Node server will login to application and show data based on group wise block chain, encrypted data, show user details based on group selection show users who left group and who joined in other groups.

Receiver User:

Receiver Using will login to application same as mobile user who will request for data and user same group public key to download no need to request if user is in same group just use public key to decrypt data and download.

LAN Networks:

Using this module LAN Networks will register with application and Login to Add Groups View Changed Users Groups View List of Groups and View Users In Each Group With Block Chain.

CHAPTER 2: SYSTEM FEASIBILITY

2.1 INTRODUCTION:

PRELIMINARY INVESTIGATION

The first and foremost strategy for development of a project starts from the thought of designing a mail enabled platform for a small firm in which it is easy and convenient of sending and receiving messages, there is a search engine address book and also including some entertaining games. When it is approved by the organization and our project guide the first activity, ie., preliminary investigation begins. The activity has three parts:

1. **Request Clarification**

2. **Feasibility Study**

3. **Request Approval**

REQUEST CLARIFICATION

After the approval of the request to the organization and project guide, with an investigation being considered, the project request must be examined to determine precisely what the system requires.

Here our project is basically meant for users within the company whose systems can be interconnected by the Local Area Network (LAN). In today's busy schedule man need everything should be provided in a readymade manner. So taking into consideration of the vastly use of the net in day to day life, the corresponding development of the portal came into existence.

FEASIBILITY ANALYSIS

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. The different feasibilities that have to be analysed are

1. **Operational Feasibility**
2. **Economic Feasibility**
3. **Technical Feasibility**

Operational Feasibility:

Operational Feasibility deals with the study of prospects of the system to be developed. This system operationally eliminates all the tensions of the admin and helps him in effectively tracking the project progress. This kind of automation will surely reduce the time and energy, which previously consumed in manual work. Based on the study, the system is proved to be operationally feasible.

Economic Feasibility:

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer-based project. As hardware was installed from the beginning & for lots of purposes thus the cost on project of hardware is low. Since the system is a network based, any number of employees connected to the LAN within that organization can use this tool from anytime. The Virtual Private Network is to be developed using the existing resources of the organization. So the project is economically feasible.

Technical Feasibility

According to Roger S. Pressman, Technical Feasibility is the assessment of the technical resources of the organization. The organization needs IBM compatible machines with a graphical web browser connected to the Internet and Intranet. The system is developed for platform independent environment. Java Server Pages, JavaScript, HTML, SQL server and WebLogic Server are used to develop the system. The technical feasibility has been carried out. The system is technically feasible for development and can be developed with the existing facility.

2.2 EXISTING SYSTEM

One of the most popular Centralized Key Management Scheme is the Logical Key Hierarchy (LKH) which was proposed by Wong et. al in and Wallner et. al in. Most of the proposed centralized protocols utilize a common TEK for all group members. The key tree in the LKH-based schemes reduces the number of messages to log(n), where n is the number of group members. These messages are required for updating the TEK whenever there are member changes. Baugher et al. defined a common architecture for multicast security key management protocols in. Sun et al. proposed a multigroup key management scheme which achieves hierarchical group access control.

2.3 DISADVANTAGES OF EXISTING SYSTEM

Most traditional key management schemes are based on a key tree, and the distribution of nodes in the key management tree mainly depends on the joining and the leaving of members over time. Besides, some key management schemes are based on a hierarchical structure, and the key manager of each hierarchy can help to manage keys more conveniently. However, the hierarchical structure may lead to extra overheads because it introduces more intermediate entity (i.e. intermediate key manager) to the key management processes.

2.4 PROPOSED SYSTEM

The emerging blockchain technique offers the potential to build a trusted, fair-minded and decentralized environment for key management scheme in wireless mobile edge networks. Besides, keeping private keys locally and storing public keys in the blockchain can reduce the overhead of rekeying process. Because when users move into a new group, their records (i.e., public keys) can be traced in the previous blocks in local or higher hierarchical blockchain, so that their identities can be verified quickly. Meanwhile, only if the quick verification is failed, the users need to generate a new key pair (i.e., rekeying) which will be appended to the blockchain. Moreover, the key pair of users will be invalid once they generate a new one, which can reduce the overhead of backward and forward secrecy in the rekeying process.

2.5 ADVANTAGES OF PROPOSED SYSTEM

This paper aims to reduce the rekeying overhead and improve the security of key management in MEC network. In order to deal with all the challenges mentioned above, we analyse the characteristics of key management schemes and blockchain technique, and then we construct a blockchain based key management scheme.

CHAPTER 3: SYSTEM ANALYSIS

3.1 THE STUDY OF THE SYSTEM

- To conduct studies and analyses of an operational and technological nature, and
- To promote the exchange and development of methods and tools for operational analysis as applied to defence problems.

Logical Design:

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modelling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems design are included. Logical design includes ER Diagrams i.e. Entity Relationship Diagrams

Physical design:

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified / authenticated, how it is processed, and how it is displayed as output. In Physical design, following requirements about the system are decided.

1. Input requirement,
2. Output requirements,
3. Storage requirements,
4. Processing Requirements,
5. System control and backup or recovery.

Put another way, the physical portion of systems design can generally be broken down into three sub-tasks:

1. User Interface Design
2. Data Design
3. Process Design

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the systems design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and a control processor. The H/S personal specification is developed for the proposed system.

3.2 INPUT & OUTPUT REPRESENTATION

Input Design:

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:

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.

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.

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.

    a. 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.

    b. Select methods for presenting information.

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

3.3 INTRODUCTION TO JAVA

Java Technology:

Java technology is both a programming language and a platform. 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.
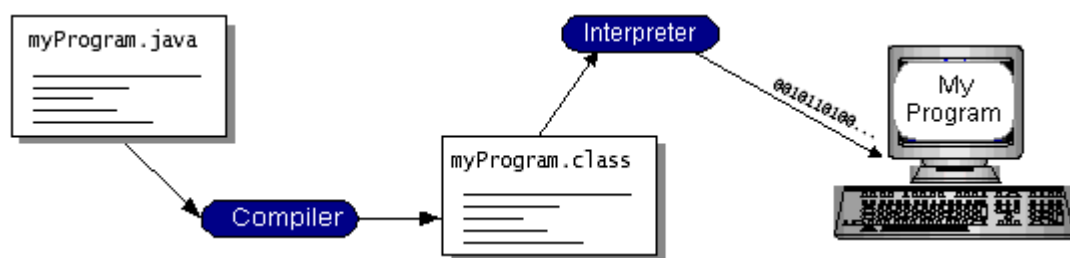
Figure 1: Working Process of Java

14

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.
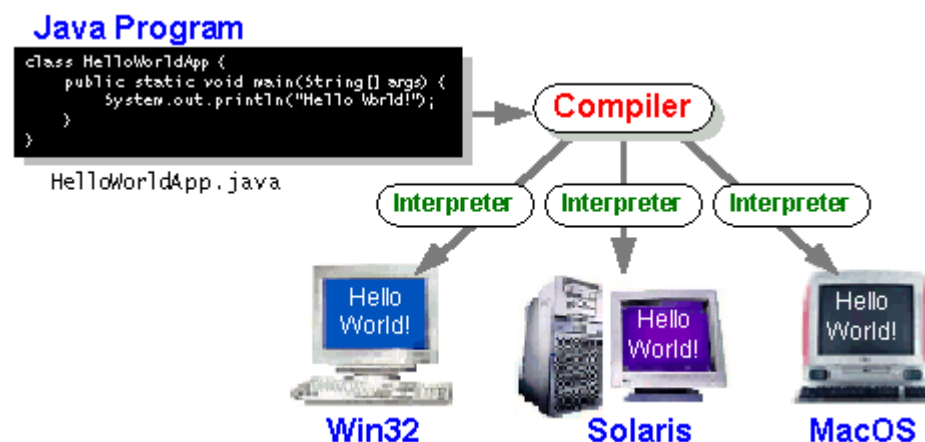
Figure 2: Working Process of Java

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.
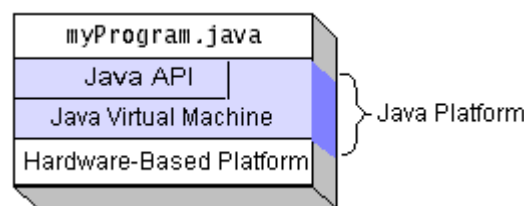
Figure 3: Java API and Virtual Machine

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.

16

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 JavaBeans, can plug into existing component architectures.
- **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC™)**: 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.
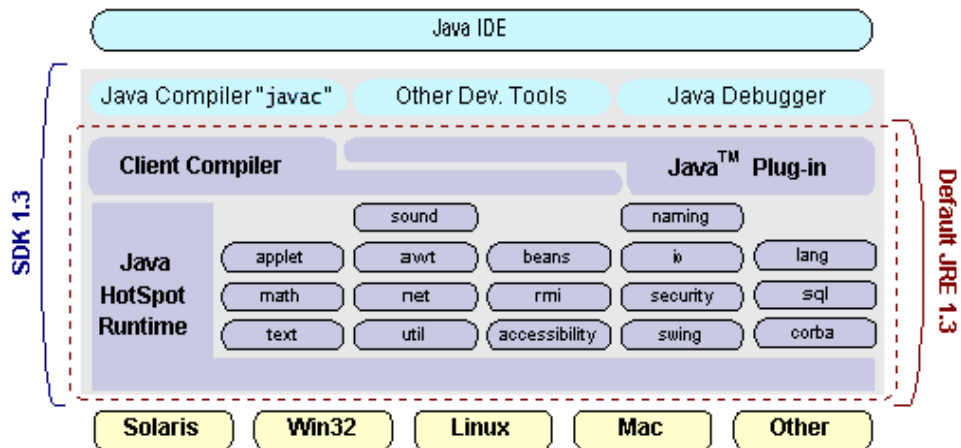
Figure 4: Java 2 SDK

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++.

18

- **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 Java^{TM} 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.

2. **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.

3. **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.

4. **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.

5. **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.

6. **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.

Networking

TCP/IP stack

The TCP/IP stack is shorter than the OSI one:
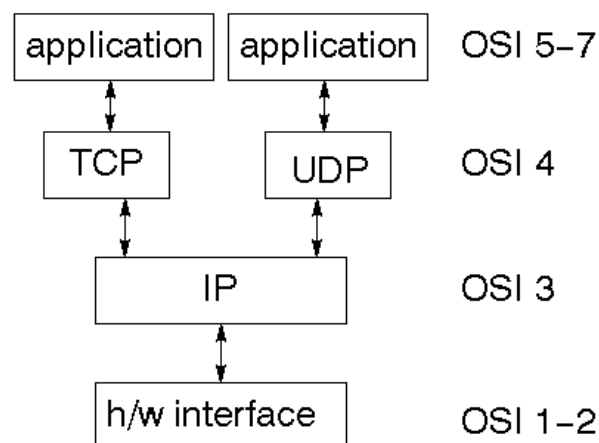
Figure 5: TCP/IP Stack

22

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 32bit 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:
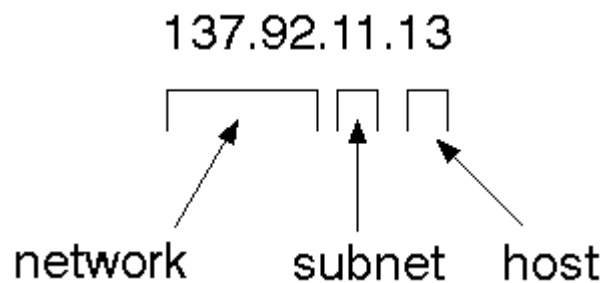
Figure 6: Total Address of TCP

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);

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:
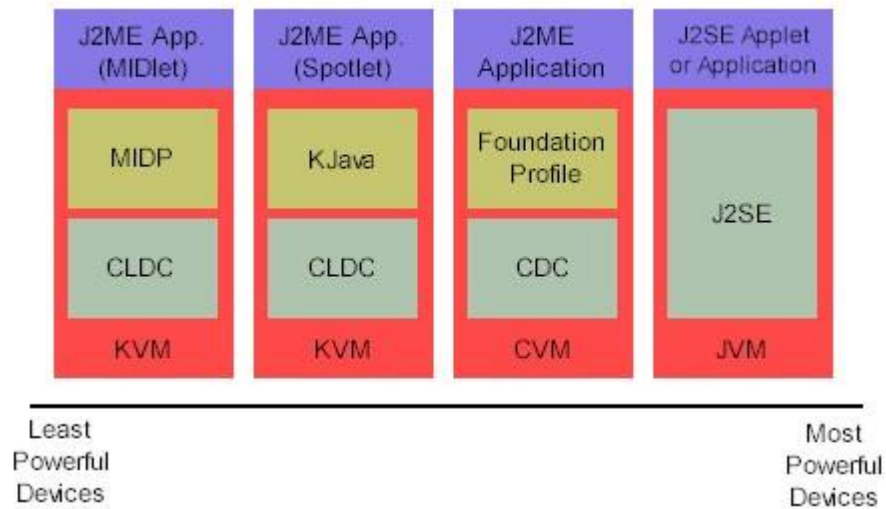
Figure 7: General J2ME Architecture

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 pre verification 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

3.4 SYSTEM REQUIREMENTS

3.4.1 HARDWARE REQUIREMENTS

System   : Pentium IV 2.4 GHz.

Hard Disk   : 200 GB.

Floppy Drive   : 1.44 Mb.

Monitor   : 15 VGA Colour.

Mouse   : Any.

Ram   : 1 GB.

3.4.2 SOFTWARE REQUIREMENTS

Operating system : Windows XP/7/10/11.

Coding Language   : Java.

Tool   : Netbeans.

Database   : MYSQL.

CHAPTER 4: SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

Architecture Flow:

Below architecture diagram represents mainly flow of request from the users to database through servers. In this scenario overall system is designed in three tiers separately using three layers called presentation layer, business layer, data link layer. This project was developed using 3-tier architecture.
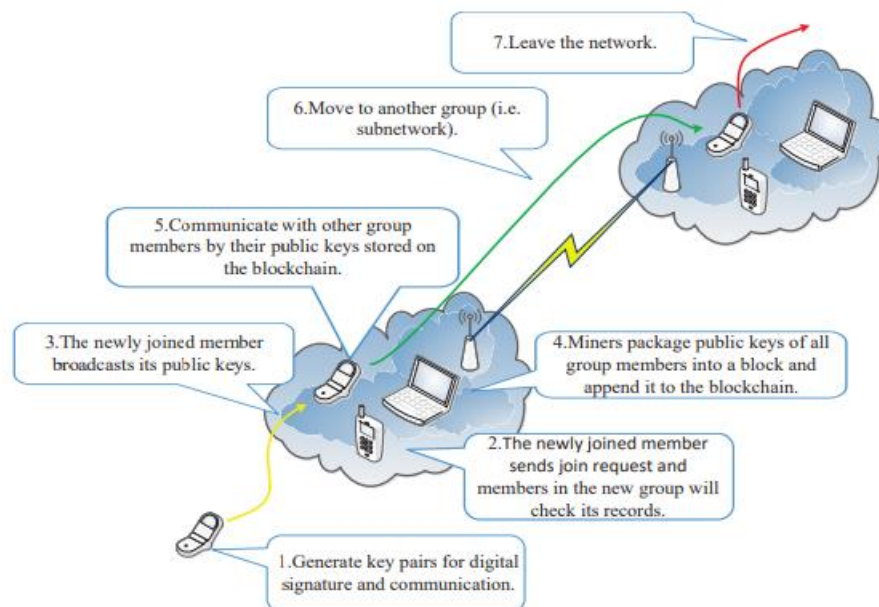
Figure 8: System Architecture Diagram

3-Tier Architecture:

The three-tier software architecture (a three-layer architecture) emerged in the 1990s to overcome the limitations of the two-tier architecture. The third tier (middle tier server) is between the user interface (client) and the data management (server) components. This middle tier provides process management where business logic and rules are executed and can accommodate hundreds of users (as compared to only 100

users with the 2-tier architecture) by providing functions such as queuing, application execution, and database staging.

The 3-tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two tier) increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user. These characteristics have made three-layer architectures a popular choice for Internet applications and net-centric information systems

Advantages of 3-Tier: Separates functionality from presentation.

- Clear separation – better understanding.
- Changes limited to well define components.
- Can be running on WWW.
- Effective network performance.

SYSTEM DESIGN

System Design Introduction:
The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

4.2 UML DIAGRAMS

Global Use Case Diagrams:
Identification of Actors:
Actor: Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical Representation:
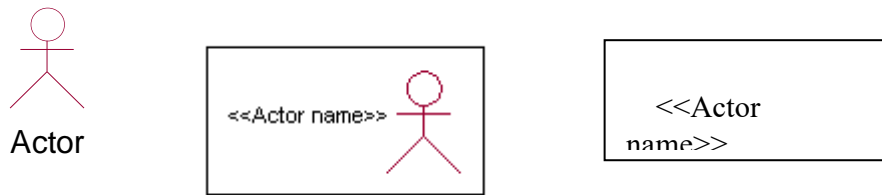
An actor is someone or something that:

Interacts with or uses the system.

- Provides input to and receives information from the system.

- Is external to the system and has no control over the use cases.

Actors are discovered by examining:

- Who directly uses the system?

- Who is responsible for maintaining the system?

- External hardware used by the system.

- Other systems that need to interact with the system.

Questions to identify actors:

- Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?

- Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.

- Which external hardware or systems (if any) use the system to perform tasks?

- What problems does this application solve (that is, for whom)?

- And, finally, how do users use the system (use case)? What are they doing with the system?

The actors identified in this system are:

a. System Administrator

b. Customer

c. Customer Care

Identification of Use cases:

Use case: A use case can be described as a specific way of using the system from a user's (actor's) perspective.

Graphical representation:
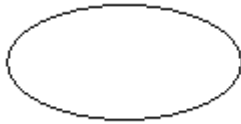
A more detailed description might characterize a use case as:

- Pattern of behaviour the system exhibits
- A sequence of related transactions performed by an actor and the system
- Delivering something of value to the actor

Use cases provide a means to:

- capture system requirements
- communicate with the end users and domain experts
- test the system

Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.

Guide lines for identifying use cases:

- For each actor, find the tasks and functions that the actor should be able to perform or that the system needs the actor to perform. The use case should represent a course of events that leads to clear goal
- Name the use cases.
- Describe the use cases briefly by applying terms with which the user is familiar.

This makes the description less ambiguous

Questions to identify use cases:

- What are the tasks of each actor?
- Will any actor create, store, change, remove or read information in the system?
- What use case will store, change, remove or read this information?
- Will any actor need to inform the system about sudden external changes?
- Does any actor need to inform about certain occurrences in the system?

- What use cases will support and maintains the system?

Flow of Events

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favourite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

- When and how the use case starts and ends
- Use case/actor interactions
- Data needed by the use case
- Normal sequence of events for the use case
- Alternate or exceptional flows

4.2.1 Construction of Use case diagrams:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

4.2.2 Sequence Diagrams:

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

4.2.3. Class Diagram:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing

the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

4.2.4 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

CHAPTER 5: TESTING AND VALIDATION

INTRODUCTION:

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands.

Testing objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say, testing is a process of executing a program with intent of finding an error.

1. A successful test is one that uncovers an as yet undiscovered error.
2. A good test case is one that has probability of finding an error, if it exists.
3. The test is inadequate to detect possibly present errors.
4. The software more or less confirms to the quality and reliable standards.

Levels of Testing:

Code testing: This examines the logic of the program. For example, the logic for updating various sample data and with the sample files and directories were tested and verified.

Specification Testing: Executing this specification starting what the program should do and how it should performe under various conditions. Test cases for various situation and combination of conditions in all the modules are tested.

Unit testing: In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module. There are some validation checks for fields also. For example, the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system.

Each Module can be tested using the following two Strategies:

1. Black Box Testing
2. White Box Testing

BLACK BOX TESTING

What is Black Box Testing?

Black box testing is a software testing techniques in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications.

In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

The above Black Box can be any software system you want to test. For example, an operating system like Windows, a website like Google ,a database like Oracle or even

your own custom application. Under Black Box Testing , you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

Black box testing - Steps

Here are the generic steps followed to carry out any type of Black Box Testing.

- Initially requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

Types of Black Box Testing

There are many types of Black Box Testing but following are the prominent ones -

- Functional testing – This black box testing type is related to functional requirements of a system; it is done by software testers.
- Non-functional testing – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements such as performance, scalability, usability.
- Regression testing – Regression testing is done after code fixes , upgrades or any other system maintenance to check the new code has not affected the existing code.

WHITE BOX TESTING

White Box Testing is the testing of a software solution's internal coding and infrastructure. It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability. White box testing is also known as clear, open, structural, and glass box testing.

It is one of two parts of the "box testing" approach of software testing. Its counter-part, black-box testing, involves testing from an external or end-user type perspective. On the other hand, White box testing is based on the inner workings of an application and revolves around internal testing. The term "white box" was used because of the see-through box concept. The clear box or white box name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "black box testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested

What do you verify in White Box Testing?

White box testing involves the testing of the software code for the following:

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of white box testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

How do you perform White Box Testing?

To give you a simplified explanation of white box testing, we have divided it into **two basic steps**. This is what testers do when testing an application using the white box testing technique:

STEP 1): UNDERSTAND THE SOURCE CODE

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

STEP 2): REATE TEST CASES AND EXECUTE

The second basic step to white box testing involves testing the application's source code for proper flow and structure. One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include manual testing, trial and error testing and the use of testing tools as we will explain further on in this article.

Sl # Test Case:	UTC-1
Name of Test:	All Users Registration Process
Items Being Tested:	Registration process
Sample Input:	Fill form enter details
Expected Output:	Registration data stored in database and validation message
Actual Output:	registration successful
Remarks:	Pass.

Table 1: Test Case 1

Sl # Test Case:	UTC-2
Name of Test:	Upload file with block chain
Items Being Tested:	Data stored with permissions
Sample Input:	Text data
Expected Output:	Data stored in cloud and database with confirmation and block chain
Actual Output:	File upload success.
Remarks:	success.

Table 2: Test Case 2

Sl # Test Case:	ITC-1
Name of Test:	Blok chain key generate
Item being tested:	Block chain Key is generated for uploaded data or not
Sample Input:	Text file data
Expected output:	Block chain must be generated
Actual output:	Key generated and given to cloud
Remarks:	Pass.

Table 3: Test Case 3

Sl # Test Case:	STC-1
Name of Test:	Join group key
Item being tested:	Each user has key to join group
Sample Input:	Group name
Expected output:	Confirmation after joining group
Actual output:	Users joined in group
Remarks:	Pass

Table 4: Test Case 4

CHAPTER 6: SCREENSHOTS

Blockchain Based Secure Key Management for Mobile Edge Computing

| Home | User | Black Chain Manager | Node Server | Receiver User | Lan Networks |

Welcome shiva

User Menu

✔ User Home
✔ User Profile
✔ Join Group
✔ Block Chain Key
✔ Upload Files
✔ Change Group
✔ View Data
✔ Logout

Blockchain Based Secure Key Management for Mobile Edge Computing

| Home | User | Black Chain Manager | Node Server | Receiver User | Lan Networks |

USER Join Group

shiva

Group1
Group1
Group2
Group3

JOIN   CANCEL

User Menu

✔ User Home
✔ User Profile
✔ Join Group
✔ Block Chain Key
✔ Upload Files
✔ Change Group
✔ View Data
✔ Logout

Blockchain Based Secure Key Management for Mobile Edge Computing

| Home | User | Black Chain Manager | Node Server | Receiver User | Lan Networks |

USER LOGIN

myraprojects@gmail.com

•••••

••••

LOGIN   CANCEL

Registration

REGISTRATION

Your Name	Password
e-mail	dd - mm - yyyy
Gender	Contact No
State	Country

Choose File   No file chosen

REGISTER   CLEAR

Sidebar Menu

✔ Home
✔ User
✔ Cloud
✔ Registration

CHAPTER 7: SAMPLE CODE

Block Chain.java

package keywordsearch;

import java.io.File;

import java.io.FileInputStream;

import java.util.Properties;


```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author java2
 */
public class BlockchainDemo {


    public static int blockchainkey(String msg) {
        int plainData = 0;
        try {
         Blockchain blockchain = new Blockchain();
        blockchain.addBlock(new String[]{msg});
       // blockchain.addBlock(new String[]{"Mohamed send 60 bitcoin to mom."});
       //  blockchain.addBlock(new String[]{"Omar send 13 bitcoin to karim."});

        System.out.println("Blocks hash:");
        for(Block block : blockchain.getBlocks()) {
           //plainData =block.getHash();
           System.out.println("block #" +blockchain.getBlocks());
```

```java
        if ( block.getIndex() == 1)
        {
           plainData = block.getHash();
           System.out.println("blockffffffffffff #" + block.getIndex()  + ": " +
plainData);
        }


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


        return plainData;


    }
}
```

Generate Block.Java

```java
package keywordsearch;



import java.util.Arrays;

public class Block {
    private int index;
    private int previousHash;
    private int hash;
    private String[] transactions;

    public Block(int index, int previousHash, String[] transactions) {
```

```java
        this.index = index;
        this.previousHash = previousHash;
        this.transactions = transactions;

        Object[] content = {
            Arrays.hashCode(transactions),
            previousHash,
            index
        };
        this.hash = Arrays.hashCode(content);
    }

    public int getIndex() {
        return index;
    }

    public int getPreviousHash() {
        return previousHash;
    }

    public int getHash() {
        return hash;
    }

    public String[] getTransactions() {
        return transactions;
    }
}
```

Databaseconnection

```java
package databaseconnection;
import java.sql.*;
```

```java
public class databasecon
{
static Connection co;
public static Connection getconnection()
{


        try
        {
                Class.forName("com.mysql.jdbc.Driver");
                co =
DriverManager.getConnection("jdbc:mysql://localhost:3306/securedata","root","root
");
        }
        catch(Exception e)
        {
                System.out.println("Database Error"+e);
        }
        return co;
}


}
```

Decryption

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package action;


/**
```

```
 *
 * @author java2
 */
import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;
import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.util.Scanner;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.swing.JOptionPane;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;

public class decryption {
//public static void main(String args[])
//{
//          Scanner s=new Scanner(System.in);
//          System.out.println("Enter encrypted Text and key");
//          String text=s.next();
//          String key=s.next();
//       new decryption().decrypt(text,key);
//}

   public String decrypt(String txt, String skey) {
      String decryptedtext = null;
      try {

         //converting string to secretkey
         byte[] bs = Base64.decode(skey);
```

```java
        SecretKey sec = new SecretKeySpec(bs, "AES");
        System.out.println("converted string to seretkey:" + sec);


        System.out.println("secret key:" + sec);


        Cipher aesCipher = Cipher.getInstance("AES");//getting AES instance
        aesCipher.init(Cipher.ENCRYPT_MODE, sec);//initiating ciper encryption
using secretkey


        byte[] byteCipherText = new BASE64Decoder().decodeBuffer(txt);
//encrypting data


        //  System.out.println("ciper text:"+byteCipherText);
        aesCipher.init(Cipher.DECRYPT_MODE, sec,
aesCipher.getParameters());//initiating ciper decryption


        byte[] byteDecryptedText = aesCipher.doFinal(byteCipherText);
        decryptedtext = new String(byteDecryptedText);


        System.out.println("Decrypted Text:" + decryptedtext);
    } catch (Exception e) {
        System.out.println(e);
    }
    return decryptedtext;
  }


  String decrypt(String str, SecretKey sec) {
      throw new UnsupportedOperationException("Not supported yet."); //To change
body of generated methods, choose Tools | Templates.
  }


}
```

Download

```
<%@page import="java.sql.*"%>
<%@page import="Distributed.Dbconnection"%>
<%@ page session="true" %>
    <%
    String username = session.getAttribute("user").toString();
    String task = request.getParameter("task");
    String location = request.getParameter("location");
    String status = null;
    String person = null;


    try{


    Connection con=Dbconnection.getConnection();
    PreparedStatement ps=con.prepareStatement("insert into task values(?,?,?,?,?)");

    ps.setString(1,username);
    ps.setString(2,task);
    ps.setString(3,location);

    ps.setString(4,person);
    ps.setString(5,status);




    int i=ps.executeUpdate();
    if(i>0)
    {
    response.sendRedirect("owner_upload.jsp?msg=Registered");
    }
```

```
else{
response.sendRedirect("ownerreg.jsp?msg1=Failed");
}
%>
<%
}


catch(Exception e)
{


  out.println(e);


}
%>
```

View task

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.PreparedStatement"%>
<%@page import="Distributed.Dbconnection"%>
<%@page import="java.sql.Connection"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Distributed Time-Sensitive Task Selection in Mobile Crowd Sensing</title>
<meta http-equiv="Content-type" content="text/html; charset=utf-8" />
<link rel="stylesheet" href="css/style.css" type="text/css" media="all" />
<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
<script type="text/javascript" src="js/jquery.jcarousel.js"></script>
<script src="js/cufon-yui.js" type="text/javascript" charset="utf-8"></script>
```

```jsp
<script src="js/Chaparral_Pro.font.js" type="text/javascript" charset="utf-8"></script>
<script type="text/javascript" src="js/jquery-func.js"></script>
<link rel="shortcut icon" type="image/x-icon" href="css/images/favicon.ico" />
</head>
    <%
if(request.getParameter("mg")!=null){%>

  <script>alert('Login Sucessfully..!')</script>
}

<%}
if(request.getParameter("msg1")!=null){%>

 <script>alert('Owner Registration Failed..!')</script>
}
<%
}
%>
<%
if(request.getParameter("m3")!=null){%>

  <script>alert('username already exists')</script>
}

<%}
if(request.getParameter("")!=null){%>

 <script>alert('Owner Registration Failed..!')</script>
}
<%
}
%>
```

```html
<body>
<!-- START PAGE SOURCE -->
<div id="header">
   <br>
  <div class="shell">
   <h1>Distributed Time-Sensitive Task Selection in Mobile Crowd Sensing</h1>
   <div class="search">


   </div>
  </div>
</div>
<div id="navigation">
  <div class="shell">
   <ul>
   <li><a  href="ownerhome.jsp">HOME</a></li>
    <li><a href="owner_upload.jsp">Upload Task</a></li>
    <li><a  class="active" href="owner_viewtask.jsp">View Task</a></li>
    <li><a href="owner_assign_reward.jsp">Assign Reward</a></li>
    <li><a href="owner_download.jsp">Delivery Status</a></li>
    <li><a href="logout.jsp">Logout</a></li>
   </ul>
  </div>
</div>
<div id="featured">
  <div class="shell">
   <div class="slider-carousel">
    <ul>
     <li>
      <div class="info">
      <p> We
```

provide a new efficient RDPC protocol based on homomorphic
hash function. The new scheme is provably secure against forgery
attack, replace attack and replay attack based on a typical

security model. To support data dynamics, an operation record table (ORT) is introduced to track operations on file blocks. We further give a new optimized implementation for the ORT which makes the cost of accessing ORT nearly constant. Moreover, we make the comprehensive performance analysis which shows that our scheme has advantages in computation and communication costs. Prototype implementation and experiments exhibit that the scheme is feasible for real applications. </p>

```
    </div>
     <div class="image"> <a href="#"><img src="css/images/1.png" alt="" /></a>
</div>
    <div class="cl"> </div>
    </li>
    <li>
     <div class="info">
      <p>We
```

provide a new efficient RDPC protocol based on homomorphic hash function. The new scheme is provably secure against forgery attack, replace attack and replay attack based on a typical security model. To support data dynamics, an operation record table (ORT) is introduced to track operations on file blocks. We further give a new optimized implementation for the ORT which makes the cost of accessing ORT nearly constant. Moreover, we make the comprehensive performance analysis which shows that our scheme has advantages in computation and communication costs. Prototype implementation and experiments exhibit that the scheme is feasible for real applications.</p>

```
    </div>
    <div class="image"> <a href="#"><img src="css/images/2.jpg" alt="" /></a>
</div>
    <div class="cl"> </div>
    </li>
```

```
        </ul>
      </div>
    </div>
</div>
<%

String username = session.getAttribute("user").toString();
    String task = request.getParameter("task");
    String location = request.getParameter("location");

    Connection con = null;
        con = Dbconnection.getConnection();
        PreparedStatement pst=con.prepareStatement("select * from taskhandler where
location = '"+location+"' ");
        ResultSet rs=pst.executeQuery();
%>
<div id="main">
  <div class="shell">
    <div id="main-boxes">

     <center>
       <p align="justify">
    <p><font color="black" size="5"> Assign Task </font></p><br/>

    <form name="myform" autocomplete="off" action="owner_viewtask2.jsp"
method="post" onsubmit="return validateform()">
    <table align="center" width="321">
    <tr>
    <td width="191" height="43"><font color="black">User Name </td>
    <td width="218"><input autocomplete="off" value="<%=username%>"
name="username" required="" placeholder="User Name" /></td>
    </tr>
    <tr>
```

```
<td height="43"><font color="black">Task </td>
<td width="218"><input type="text" value="<%=task%>" name="task"
required="" placeholder="Password" /></td>
</tr>
<tr>
<td height="43"><font color="black"> Location</td>
<td><input name="location" autocomplete="off" value="<%=location%>"
required="" placeholder="Email ID"/></td>
</tr>


<tr>
<td height="43"><font color="black">Select Task Handler</td>
<td><select name="th" style="width:170px;" required="">
<option>--Select--</option>
<%
while(rs.next()){
%>


<option><%=rs.getString("username")%></option>


<%
}
%>
</select></td>
</tr>




<tr>
<td height="43" rowspan="3">
```

```html
    <p> </p></td>
    <td align="left" valign="middle"> <p> 
    </p>
    <p>
    <input name="submit" type="submit" value="Assign" />
    </p>
    <div align="right">
    </div></td>
    </tr>
    </table>
    </form>
    </p>


     </center>




    </div>



    <div class="cl"> </div>
  </div>
</div>
<div class="footer">
  <div class="shell">
    <p class="rf"></a></p>
    <div style="clear:both;"></div>
  </div>
</div>
<script type="text/javascript">pageLoaded();</script>
<!-- END PAGE SOURCE -->
</body>
</html>
```

Task handler:

```jsp
<%@page import="java.util.Random"%>
<%@page import="java.sql.*"%>
<%@page import="Distributed.Dbconnection"%>
<%@ page session="true" %>
<%@page import="Distributed.Mail"%>
<%
    String username = request.getParameter("username");
    String password = request.getParameter("password");


    try{


    Connection con=Dbconnection.getConnection();
    Random s = new Random();
    int otp = s.nextInt(10000 - 5000) +25000 ;


    Statement st = con.createStatement();
    ResultSet rs = st.executeQuery("select * from taskhandler where email=
'"+username+"' and password='"+password+"'");
    if(rs.next())
    {


    String user = rs.getString("username");
    session.setAttribute("user",user);
    String email = rs.getString("email");
    session.setAttribute("email",email);



    response.sendRedirect("taskhandlerhome.jsp?m1");
```

```
}
else
{
response.sendRedirect("taskhandler.jsp?msg1=LoginFail");
}
}
catch(Exception e)
{
System.out.println("Error in emplogact"+e.getMessage());
}
%>
```

CHAPTER 8: CONCLUSION

We proposed a blockchain-based key management scheme that enables mobile devices in MEC to communicate securely while enjoying the flexibility of moving between subnetworks. In addition to avoiding single point of attacks, the proposed scheme can minimize the computation, communication, and storage overheads, that are associated with key generation, key distribution and key storage respectively. We performed a detailed security analysis of the proposed scheme based on the attack to demonstrate its security strength in a mobile environment. Experimental results are undertaken to show that the proposed scheme significantly outperforms the existing works DKM (in terms of computation overhead) and TKM (in terms of computation and communication overhead). Finally, utilizing the deep reinforcement learning to train the optimal parameters will be the future work.

CHAPTER 9: FUTURE SCOPE

Blockchain-based secure key management offers a promising future for mobile edge computing. Its decentralized infrastructure enhances security by distributing encryption keys across a network, reducing single points of failure. This ensures data and key integrity, safeguarding against tampering. Additionally, blockchain's role in identity management allows for secure user and device authentication and access control. The technology's transparent and auditable ledger improves data verification and trustworthiness, while interoperability facilitates seamless interactions and secure resource sharing across different systems. Smart contracts automate key management tasks, increasing efficiency and reducing errors. Blockchain's scalability and flexibility enable it to adapt to the growing demands of mobile edge computing. Privacy protection through techniques like zero-knowledge proofs enhances data control, and adherence to compliance and regulatory standards is supported by blockchain's immutable ledger. Ongoing research and innovation in this field continue to drive the development of new solutions for emerging challenges.

CHAPTER 10: REFERENCES

[1] (2019). *Wikipedia: E-coupon*. [Online]. Available: https://en.wikipedia. org/wiki/E-coupon

[2] C. Blundo, S. Cimato, and A. De Bonis, ``Secure E-coupons,'' *Electron. Commerce Res.*, vol. 5, no. 1, pp. 117_139, Jan. 2005. [3] (2016). *World Mobile Coupons Market to Grow at 73.1% CAGR to 2020*. [Online]. Available: https://www.prnewswire.com/news-releases/world-mobile-coupons-market-to% -grow-at-7314-cagr-to-2020-603320306.html

[4] (2017). *Coupon Fraud is Crime, Even if it Feels Harmless: Coupon Coun-selor*. [Online]. Available: https://goo.gl/2emab1.

[5] S.-C. Hsueh and J.-H. Zeng, ``Mobile coupons using blockchain technol-ogy,'' in *Proc. Int. Conf. Intell. Inf. Hiding Multimedia Signal Process.* Springer, 2018, pp. 249_255.

[6] A. Knight and N. Dai, ``Objects and the web,'' *IEEE Softw.*, vol. 19, no. 2, pp. 51_59, Mar. 2002.

[7] (2018). *Quorum*. [Online]. Available: https://github.com/jpmorganchase/ quorum

[8] (2017). *Coupon Statistics: The Ultimate Collection*. [Online]. Available: https://blog.accessdevelopment.com/ultimate-collection-coupon-statistic%s

[9] (2017). emphDigital Coupon Marketing_Statistics and Trends. [Online]. Available: https://www.invespcro.com/blog/digital-coupon-marketing

[10] (2019). *Digital Coupons Continue to be the Fastest Growing Method of Redemption due to Shoppers' Increased Demand for Convenience.* [Online]. Available: https://www.globenewswire.com/news-release/2019/ 02/13/1724510/0/en/Digi%tal-Coupons-Continue-to-be-the-Fastest-Growing-Method-of-Redemption-Due-to-Sho%ppers-Increased-Demand-for-Convenience.html

[11] (2017). *The Coupon Insider: Digital vs. Paper Coupons*. [Online]. Available: https://livingonthecheap.com/coupon-insider-digital-paper-coupons//

[12] R. G.-P. M.-V. Agarwal and N. Modani, ``An architecture for secure generation and veri_cation of electronic coupons,'' in *Proc. USENIX Annu. Tech. Conf.*, Boston, MA, USA, Jun. 2001, p. 51.

[13] S.-C. Hsueh and J.-M. Chen, ``Sharing secure m-coupons for peer-generated targeting via eWOM communications,'' *Electron. Commerce Res. Appl.*, vol. 9, no. 4, pp. 283_293, Jul. 2010.

[14] R. Rivest, ``The MD5 message-digest algorithm,'' Tech. Rep., 1992.

[15] C.-C. Chang, C.-C. Wu, and I.-C. Lin, ``A secure e-coupon system for mobile users,'' *Int. J. Comput. Sci. Netw. Secur.*, vol. 6, no. 1, p. 273, 2006.

[16] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, ``Blockchain technology: Beyond bitcoin,'' *Appl. Innov.*, vol. 2, nos. 6_10, p. 71, 2016.

[17] S. Nakamoto, ``Bitcoin: A peer-to-peer electronic cash system,'' Tech. Rep., 2008.

[18] M. Szydlo, ``Merkle tree traversal in log space and time,'' in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Springer, 2004, pp. 541_554.

[19] M. Castro and B. Liskov, ``Practical Byzantine fault tolerance,'' in *Proc. OSDI*, vol. 99, 1999, pp. 173_186.

[20] N. Szabo, ``Smart contracts: Building blocks for digital markets,'' Tech. Rep., 2018.

[21] V. Buterin, ``A next-generation smart contract and decentralized application platform,'' Tech. Rep., 2014.