CHAPTER 1

INTRODUCTION

1.1 PROJECT INTRODUCTION

Cloud computing is a new computing paradigm that is built on virtualization, parallel and distributed computing, utility computing, and service-oriented architecture. In the last several years, cloud computing has emerged as one of the most influential paradigms in the IT industry, and has attracted extensive attention from both academia and industry. Cloud computing holds the promise of providing computing as the fifth utility after the other four utilities (water, gas, electricity, and telephone). The benefits of cloud computing include reduced costs and capital expenditures, increased operational efficiencies, scalability, flexibility, immediate time to market, and so on. Different service-oriented cloud computing models have been proposed, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Numerous commercial cloud computing systems have been built at different levels, e.g., Amazon's EC2, Amazon's S3, and IBM's Blue Cloud are IaaS systems, while Google App Engine and Yahoo Pig are representative PaaS systems, and Google's Apps and Sales force's Customer Relation Management (CRM) System belong to SaaS systems. With these cloud computing systems, on one hand, enterprise users no longer need to invest in hardware/software systems or hire IT professionals to maintain these IT systems, thus they save cost on IT infrastructure and human resources; on the other hand, computing utilities provided by cloud computing are being offered at a relatively low price in a pay-as-you-use style. For example, Amazon's S3 data storage service with 99.99% durability charges only \$0.06 to \$0.15 per gigabyte-month, while traditional storage cost ranges from \$1.00 to \$3.50 per gigabyte-month according to Zetta Inc. . Although the great benefits brought by cloud computing paradigm are exciting for IT companies, academic researchers, and potential cloud users, security problems in cloud computing become serious obstacles which, without being appropriately addressed, will prevent cloud computing extensive applications and usage in the future. One of the prominent security concerns is data security and privacy in cloud computing due to its Internet- based data storage and management. In cloud computing, users have to give up their data to the cloud service provider for storage and business operations, while the cloud service provider is usually a commercial enterprise which cannot be totally trusted. Data represents an extremely important asset for any organization, and enterprise users will face serious consequences if its confidential data is disclosed to their business competitors or the public. Thus, cloud users in the first place want to make sure that their data are kept confidential to outsiders, including the cloud provider and their potential competitors. This is the first data security requirement. Data confidentiality is not the only security requirement. Flexible and fine-grained access control is also strongly desired in the service-oriented cloud computing model. A health-care information system on a cloud is required to restrict access of protected medical records to eligible doctors and a customer relation management system running on a cloud may allow access of customer information to high-level executives of the company only. In these cases, access control of sensitive data is either required by legislation (e.g., HIPAA) or company regulations.

Access control is a classic security topic which dates back to the 1960s or early 1970s, and various access control models have been proposed since then. Among them, Bell-La Padula (BLP) and BiBa are two famous security models. To achieve flexible and fine-grained access control, a number of schemes have been proposed more recently. Unfortunately, these schemes are only applicable to systems in which data owners and the service providers are within the same trusted domain. Since data owners and service providers are usually not in the same trusted domain in cloud computing, a new access control scheme employing attributed-based encryption is proposed by Yu *e*t al., which adopts the so-called key-policy attribute-based encryption (KP-ABE) to enforce fine-grained access control. However, this scheme falls short of flexibility in attribute management and lacks scalability in dealing with multiple-levels of attribute authorities. We note that in contrast to KP-ABE, ciphertext-policy ABE (CP-ABE) turns out to be well suited for access control due to its expressiveness in describing access control policies. In this paper, we propose a hierarchical attribute-set-based encryption (HASBE) scheme for access control in cloud computing. HASBE extends the ciphertext-policy attribute- set-based encryption (CP-ASBE, or ASBE for short) scheme by Bobba *e*t al.

with a hierarchical structure of system users, so as to achieve scalable, flexiblem and fine-grained access control. The contribution of the paper is multifold. First, we show how HASBE extends the ASBE algorithm with a hierarchical structure to improve scalability and flexibility while at the same time inherits the feature of fine-grained access control of ASBE. Second, we demonstrate how to implement a full-fledged access control scheme for cloud computing based on HASBE. The scheme provides full support for hierarchical user grant, file creation, file deletion, and user revocation in cloud computing. Third, we formally prove the security of the proposed scheme based on the security of the CP-ABE scheme by Bethencourt *e*t al. and analyze its performance in terms of computational overhead. Lastly, we implement HASBE and conduct comprehensive experiments for performance evaluation, and our experiments demonstrate that HASBE has satisfactory performance.

1.2 OVERVIEW

To achieve flexible and fine-grained access control, a number of schemes have been proposed more recently. Unfortunately, these schemes are only applicable to systems in which data owners and the service providers are within the same trusted domain. Since data owners and service providers are usually not in the same trusted domain in cloud computing, a new access control scheme employing attributed-based encryption is proposed, which adopts the so-called key-policy attribute-based encryption (KP-ABE) to enforce fine-grained access control. However, this scheme falls short of flexibility in attribute management and lacks scalability in dealing with multiple-levels of attribute authorities.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Our existing solution applies cryptographic methods by disclosing data decryption keys only to authorize users. These solutions inevitably introduce a heavy computation overhead on the data owner for key distribution and data management when fine grained data access control is desired, and thus do not scale well.

2.2 DISADVANTAGES OF EXISTING SYSTEM

Software update/patches - could change security settings, assigning privileges too low, or even more alarmingly too high allowing access to your data by other parties.

Security concerns - Experts claim that their clouds are 100% secure - but it will not be their head on the block when things go awry. It's often stated that cloud computing security is better than most enterprises. Also, how do you decide which data to handle in the cloud and which to keep to internal systems once decided keeping it secure could well be a full-time task?

Control - Control of your data/system by third-party. Data - once in the cloud always in the cloud! Can you be sure that once you delete data from your cloud account will it not exist any more... ...or will traces remain in the cloud

2.3 LITERATURE SURVEY

2.3.1 about the Domain

Cloud computing is a new computing paradigm that is built on virtualization, parallel and distributed computing, utility computing, and service-oriented architecture. In the last several years, cloud computing has emerged as one of the most influential paradigms in the IT industry, and has attracted extensive attention from both academia and industry. Cloud computing holds the promise of providing computing as the fifth utility after the other four utilities (water, gas, electricity, and telephone). The benefits of cloud computing include reduced costs and capital expenditures, increased operational efficiencies, scalability, flexibility, immediate time to market, and so on. Different service-oriented cloud computing models have been proposed, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Numerous commercial cloud computing systems have been built at different levels, e.g., Amazon’s EC2, Amazon’s S3, and IBM’s Blue Cloud are IaaS systems, while Google App Engine and Yahoo Pig are representative PaaS systems, and Google’s App and Salesforce’s Customer Relation Management (CRM) System belong to SaaS systems. With these cloud computing systems, on one hand, enterprise users no longer need to invest in hardware/software systems or hire IT professionals to maintain these IT systems, thus they save cost on IT infrastructure and human resources; on the other hand, computing utilities provided by cloud computing are being offered at a relatively low price in a pay-as-you-use style.

2.3.2 Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility

AUTHORS: Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic.

With the significant advances in Information and Communications Technology (ICT) over the last half century, there is an increasingly perceived vision that computing will one day be the 5th utility (after water, electricity, gas, and telephony). This computing utility, like all other four existing utilities, will provide the basic level of computing service that is considered essential to meet the everyday needs of the general community. To deliver this vision, a number of computing paradigms have been proposed, of which the latest one is known as Cloud computing. Hence, in this paper, we define Cloud computing and provide the architecture for creating Clouds with market-oriented resource allocation by leveraging technologies such as Virtual Machines (VMs). We also provide insights on market-based resource management strategies that encompass both customer-driven service management and computational risk management to sustain Service Level Agreement (SLA)-oriented resource allocation. In addition, we reveal our early thoughts on interconnecting Clouds for dynamically creating global Cloud exchanges and markets. Then, we present some representative Cloud platforms, especially those developed in industries, along with our current work towards realizing market-oriented resource allocation of Clouds as realized in Aneka enterprise Cloud technology. Furthermore, we highlight the difference between High Performance Computing (HPC) workload and Internet-based services workload. We also describe a meta-negotiation infrastructure to establish global Cloud exchanges and markets, and illustrate a case study of harnessing ‘Storage Clouds’ for high performance content delivery. Finally, we conclude with the need for convergence of competing IT paradigms to deliver our 21st century vision.

2.3.3 Methods and limitations of security policy reconciliation

AUTHORS: MDaniel, P. ,Prakash, A.

A security policy is a means by which participant session requirements are specified. However, existing frameworks provide limited facilities for the automated reconciliation of participant policies. This paper considers the limits and methods of reconciliation in a general-purpose policy model. We identify an algorithm for efficient two-policy reconciliation, and show that, in the worst-case, reconciliation of three or more policies is intractable. Further, we suggest efficient heuristics for the detection and resolution of intractable reconciliation. Based upon the policy model, we describe the design and implementation of the Ismene policy language. The expressiveness of Ismene, and indirectly of our model, is demonstrated through the representation and exposition of policies supported by existing policy languages. We conclude with brief notes on the integration and enforcement of Ismene policy within the Antigone communication system.

2.3.4 A unified scheme for resource protection in automated trust negotiation

AUTHORS: Ting Yu , Winslett, M.

Automated trust negotiation is an approach to establishing trust between strangers through iterative disclosure of digital credentials. In automated trust negotiation, access control policies play a key role in protecting resources from unauthorized access. Unlike in traditional trust management systems, the access control policy for a resource is usually unknown to the party requesting access to the resource, when trust negotiation starts. The negotiating parties can rely on policy disclosures to learn each other's access control requirements. However a policy itself may also contain sensitive information. Disclosing policies' contents unconditionally may leak valuable business information or jeopardize individuals' privacy. In this paper we propose UniPro, a unified scheme to model protection of resources, including policies, in trust negotiation. UniPro improves on previous work by modeling policies as first-class resources, protecting them in the same way as other resources, providing fine-grained control over policy disclosure, and clearly distinguishing between policy disclosure and policy satisfaction, which gives users more flexibility in expressing their authorization requirements. We also show that UniPro can be used with practical negotiation strategies without jeopardizing autonomy in the choice of strategy, and present criteria under which negotiations using UniPro are guaranteed to succeed in establishing trust.

2.3.5 Ciphertexts-policy attribute based encryption

AUTHORS: John Bethencourt, Amit Sahai, Brent Waters

In several distributed systems a user should only be able to access data if a user posses a certain set of credentials or attributes. Currently, the only method for enforcing such policies is to employ a trusted server to store the data and mediate access control. However, if any server storing the data is compromised, then the confidentiality of the data will be compromised. In this paper we present a system for realizing complex access control on encrypted data that we call Ciphertexts-Policy Attribute-Based Encryption. By using our techniques encrypted data can be kept confidential even if the storage server is untrusted; moreover, our methods are secure against collusion attacks. Previous Attribute- Based Encryption systems used attributes to describe the encrypted data and built policies into user's keys; while in our system attributes are used to describe a user's credentials, and a party encrypting data determines a policy for who can decrypt. Thus, our methods are conceptually closer to traditional access control methods such as Role-Based Access Control (RBAC). In addition, we provide an implementation of our system and give performance measurements.

2.3.6 Fuzzy identity based encryption

AUTHORS: Amit Sahai , Brent R. Waters

We introduce a new type of Identity Based Encryption (IBE) scheme that we call Fuzzy Identity Based Encryption. A Fuzzy IBE scheme allows for a private key for an identity id to decrypt a Ciphertexts encrypted with identity id # if and only if the identities id and id # are close to each other as measured by some metric (e.g. Hamming distance). A Fuzzy IBE scheme can be applied to enable encryption using biometric measurements as identities. The error-tolerance of a Fuzzy IBE scheme is precisely what allows for the use of biometric identities, which inherently contain some amount of noise during each measurement.

2.4 PROPOSED SYSTEM

In order to achieve secure, scalable and fine-grained access control on outsourced data in the cloud, we utilize and uniquely combine the advanced cryptographic techniques.

2.4.1 ADVANTAGES OF PROPOSED SYSTEM:

- Low initial capital investment
- Shorter start-up time for new services
- Lower maintenance and operation costs
- Higher utilization through virtualization
- Easier disaster recovery

More specifically, we associate each data file with a set of attributes, and assign each user an expressive access structure which is defined over these attributes. To enforce this kind of access control, we utilize KP-ABE to escort data encryption keys of data files. Such construction enables us to immediately enjoy fine-grandness of access control. However, this construction, if deployed alone, would introduce heavy computation overhead and cumbersome online burden towards the data owner, as he is in charge of all the operations of data/user management. Specifically, such an issue is mainly caused by the operation of user revocation, which inevitably requires the data owner to re-encrypt all the data files accessible to the leaving user, or even needs the data owner to stay online to update secret keys for users. To resolve this challenging issue and make the construction suitable for cloud computing, we uniquely combine PRE with KP-ABE and enable the data owner to delegate most of the computation intensive operations to Cloud Servers without disclosing the underlying file contents. Such a construction allows the data owner to control access of his data files with a minimal overhead in terms of computation effort and online time, and thus fits well into the cloud environment. Data confidentiality is also achieved since Cloud Servers are not able to learn the plaintext of any data file in our construction. For further reducing the computation overhead on Cloud Servers and thus saving the data owner's investment, we take advantage of the lazy re-encryption technique and allow Cloud Servers to "aggregate "computation tasks of multiple system operations. As we will discuss in section V-B, the computation complexity on Cloud Servers is either proportional to the number of system attributes, or linear to the size of the user access structure/tree, which is independent to the number of users in the system. Scalability is thus achieved. In addition, our construction also protects user access privilege information against Cloud Servers. Accountability of user secret key can also be achieved by using an enhanced scheme of KP-ABE.

2.5 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

1. Economical Feasibility
2. Technical Feasibility
3. Social Feasibility

2.5.1 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.

2.5.2 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.

2.5.3 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

CHAPTER 3

MODULES

3.1 MODULES NAME

- Data Owner Module
- Data Consumer Module
- Cloud Server Module
- Attribute based key generation Module

3.2 MODULE DESCRIPTION

3.2.1 Data Owner Module

In this module, the data owner uploads their data in the cloud server. For the security purpose the data owner encrypts the data file and then store in the cloud. The data owner can change the policy over data files by updating the expiration time. The Data owner can have capable of manipulating the encrypted data file. And the data owner can set the access privilege to the encrypted data file.

3.2.2 Data Consumer Module

In this module, the user can only access the data file with the encrypted key if the user has the privilege to access the file. For the user level, all the privileges are given by the Domain authority and the Data users are controlled by the Domain Authority only. Users may try to access data files either within or outside the scope of their access privileges, so malicious users may collude with each other to get sensitive files beyond their privileges.

3.2.3 Cloud Server Module

The cloud service provider manages a cloud to provide data storage service. Data owners encrypt their data files and store them in the cloud for sharing with data consumers.

To access the shared data files, data consumers download encrypted data files of their interest from the cloud and then decrypt them.

3.2.4 Attribute based key generation Module

The trusted authority is responsible for generating and distributing system parameters and root master keys as well as authorizing the top-level domain authorities. A domain authority is responsible for delegating keys to subordinate domain authorities at the next level or users in its domain. Each user in the system is assigned a key structure which specifies the attributes associated with the user's decryption key. The trusted authority calls the algorithm to create system public parameters PK and master key MK. PK will be made public to other parties and MK will be kept secret. When a user sends request for data files stored on the cloud, the cloud sends the corresponding ciphertexts to the user. The user decrypts them by first calling decrypt (CT, SK) to obtain DEK and then decrypt data files using DEK.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system do and not how it should be implemented.

• System	:	Pentium IV 2.4 GHz.
• Hard Disk	:	40 GB.
• Floppy Drive	:	1.44 Mb.
• Monitor	:	15 VGA Colour.
• Mouse	:	Logitech.
• Ram	:	512 Mb.

4.2 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

• Operating system	:	Windows XP Professional or windows 7
• JDK	:	1.5/ 1.6 and above
• Front End	:	JAVA, Swing(JFC),
• Database	:	My SQL

- Tool : Netbeans 7
- Application Server : Tomcat5.0/6.X

4.3 FUNCTIONAL REQUIREMENTS

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs (see also software). Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases.

4.4 NON FUNCTIONALREQUIREMENTS

Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do <requirement>", while non-functional requirements are "system shall be <requirement>". The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture.

CHAPTER 5

DATABASE DESIGN

5.1 INTRODUCTION

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

5.2 USER LOGIN/REGISTRATION

Description: This table is used for user register details. when user is logged in by using their user name and password, then our application will check the authentication whether the user exist or not in our table. In this table, three columns and six rows are created.

COLUMN Name	Data Type	Length
userid	varchar	255
passid	Int	100
Username	varchar	230
Email	varchar	210
Adds	varchar	200
Mob	Int	200

Table 5.1: User Login/Registration

5.3 STORAGE

Description: This table is used for user file storage details. In this table three columns and one row are created.

COLUMN Name	Data Type	Length
Path	Varchar	255

Table 5.2: Storage

5.4 FILE UPLOAD

Description: This table is used for file upload details. In this table three columns and four rows are created.

COLUMN Name	Data Type	Length
Fileid	Int	255
Fname	Varchar	200
Dateofupload	Varchar	200
Donwername	Varchar	200

Table 5.3: File Upload

CHAPTER 6

SYSTEM DESIGN

6.1 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the “flow” of data through an information system. It differs from the flowchart as it shows the data flow instead of the control flow of the program. A data flow diagram can also be used for the visualization of data processing.

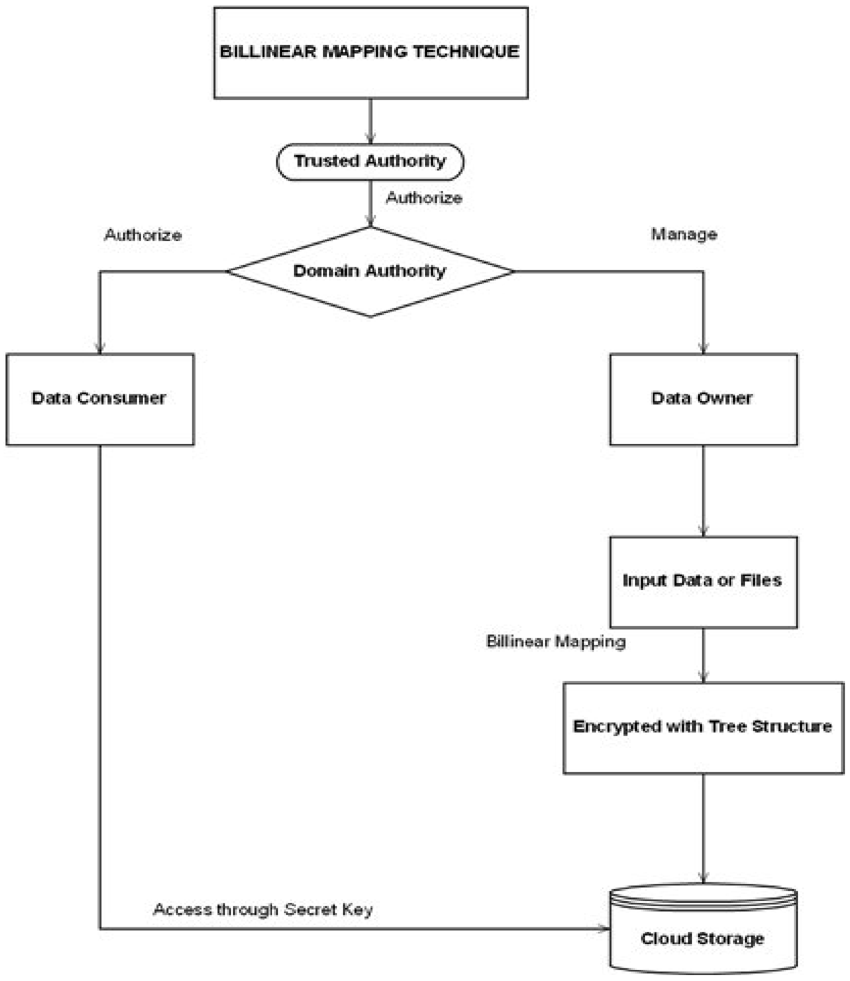


Fig. 6.1 DFD All Level

Level 1:

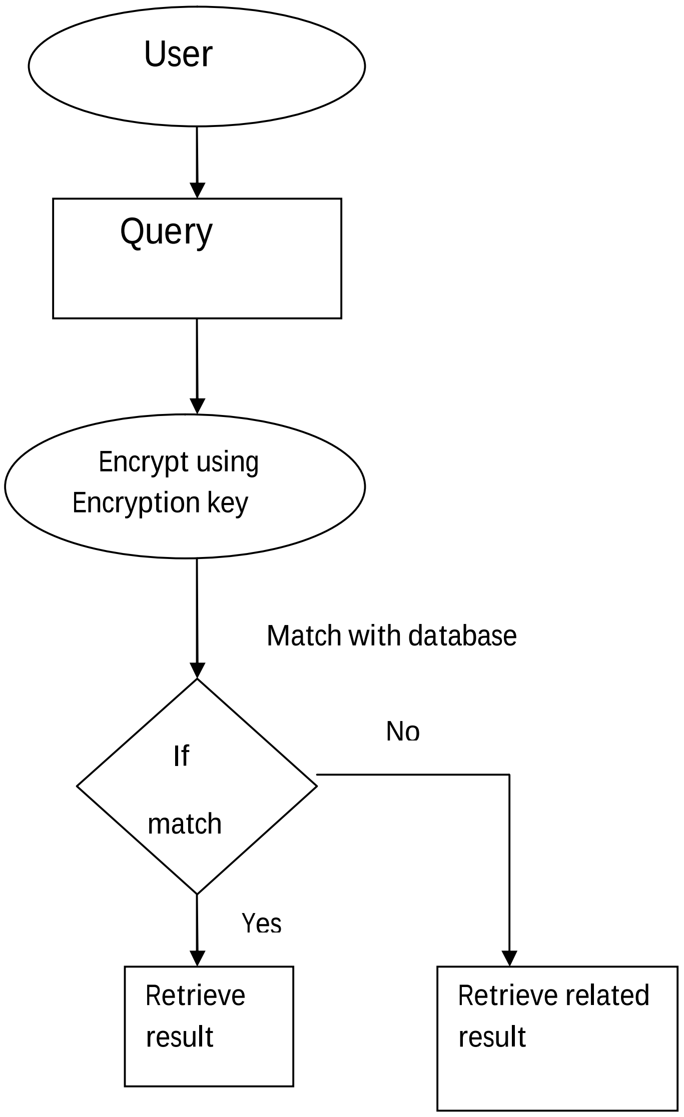


Fig. 6.2 DFD Level 1

Level 2:

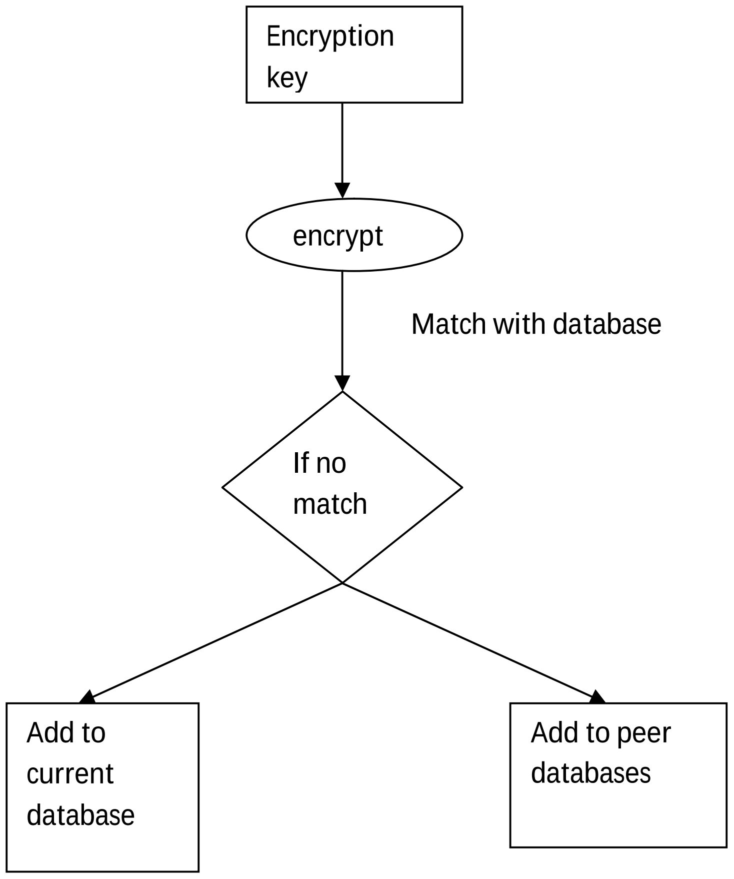


Fig. 6.3 DFD Level 2

6.2 SEQUENCE DIAGRAM

A sequence diagram in UML is a kind of interaction diagram that shows how processes operate with one another and in what order.

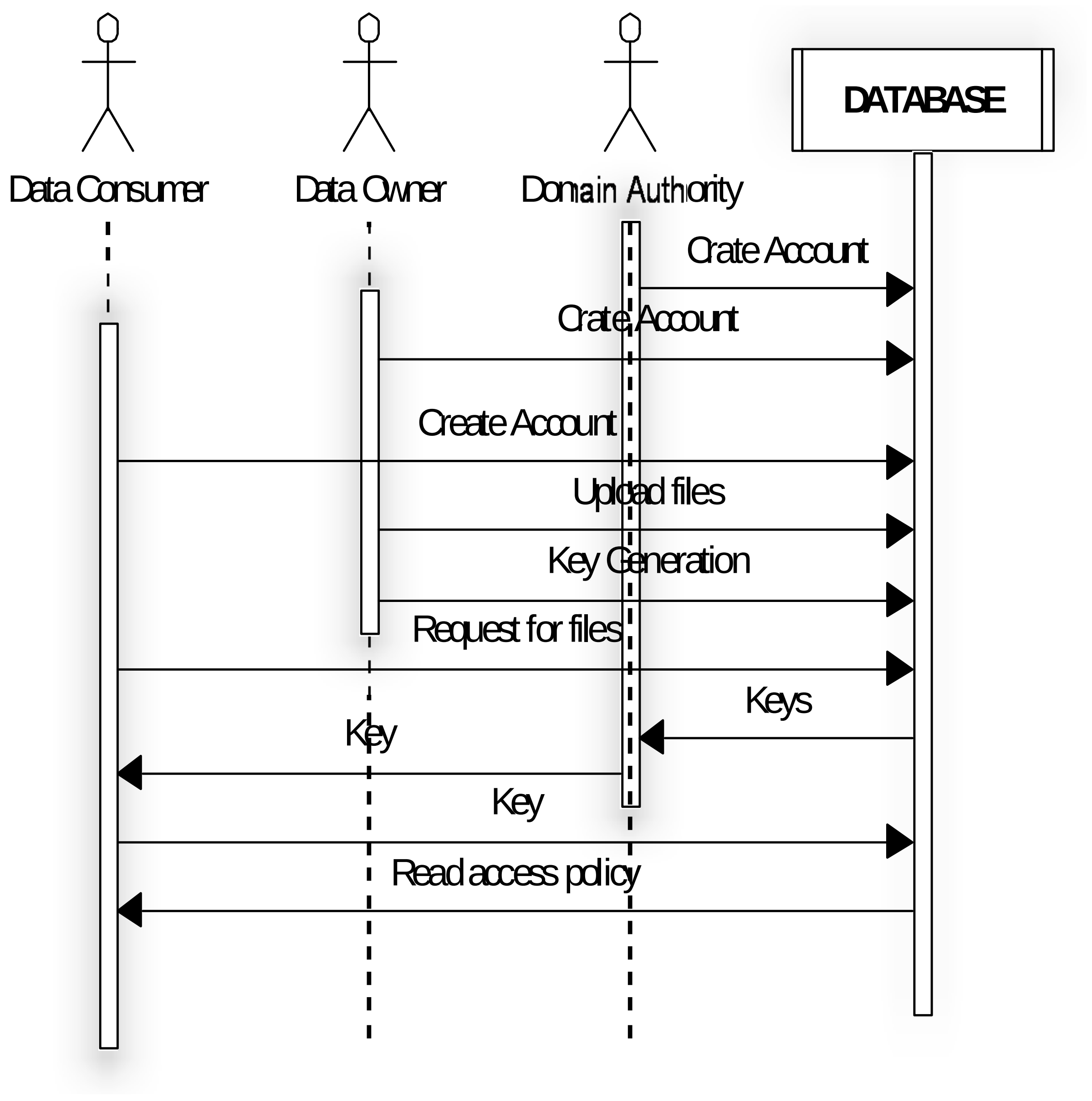


Fig. 6.4 Sequence Diagram

6.3 ACTIVITY DIAGRAM

Activity diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. UML, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system.

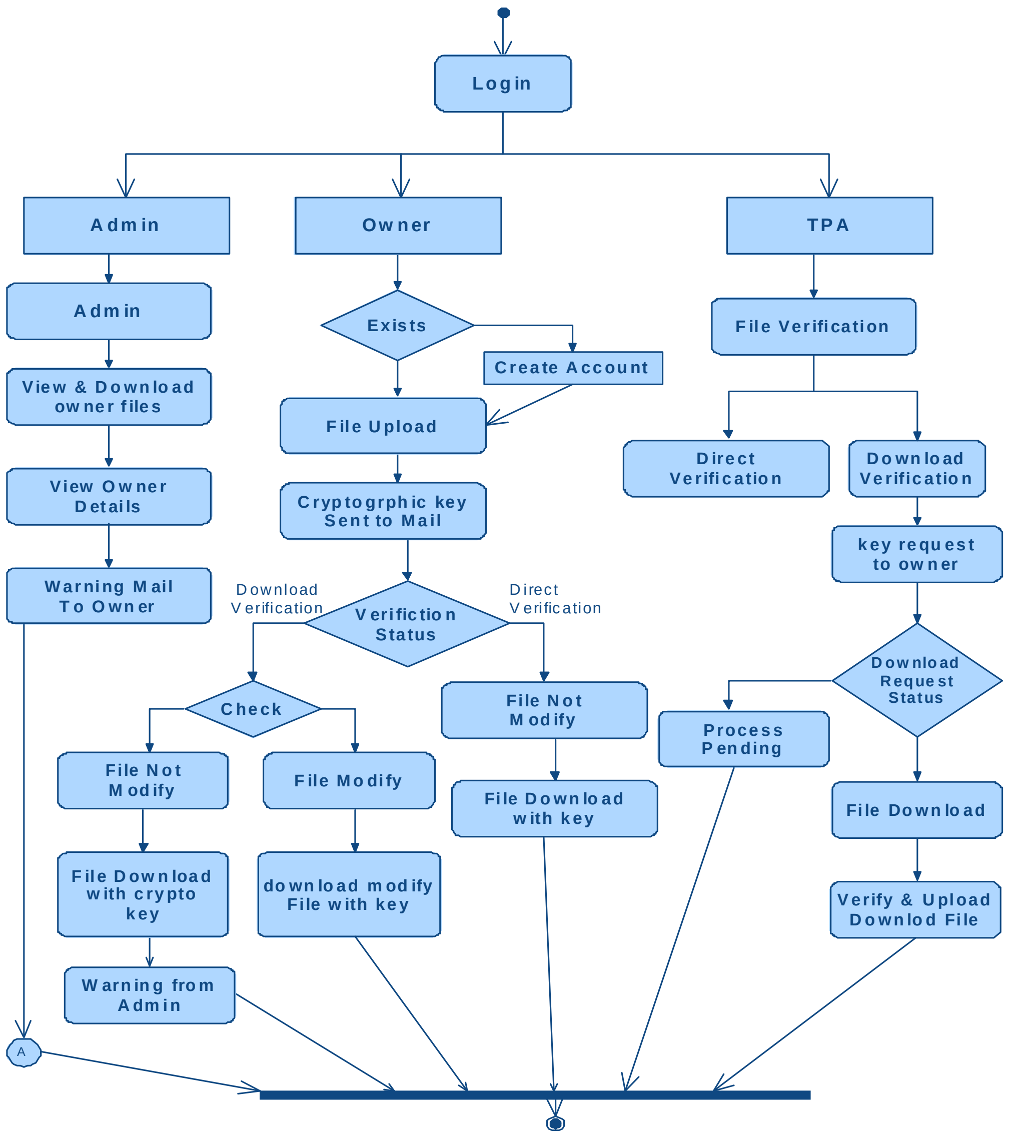


Fig. 6.5 Activity Diagram

6.4 USE CASE DIAGRAM

A use case diagram is a graph of actors, a set of use cases enclosed by a system boundary, communication (participation) association between the actors and the use cases, and generalization among the use cases.

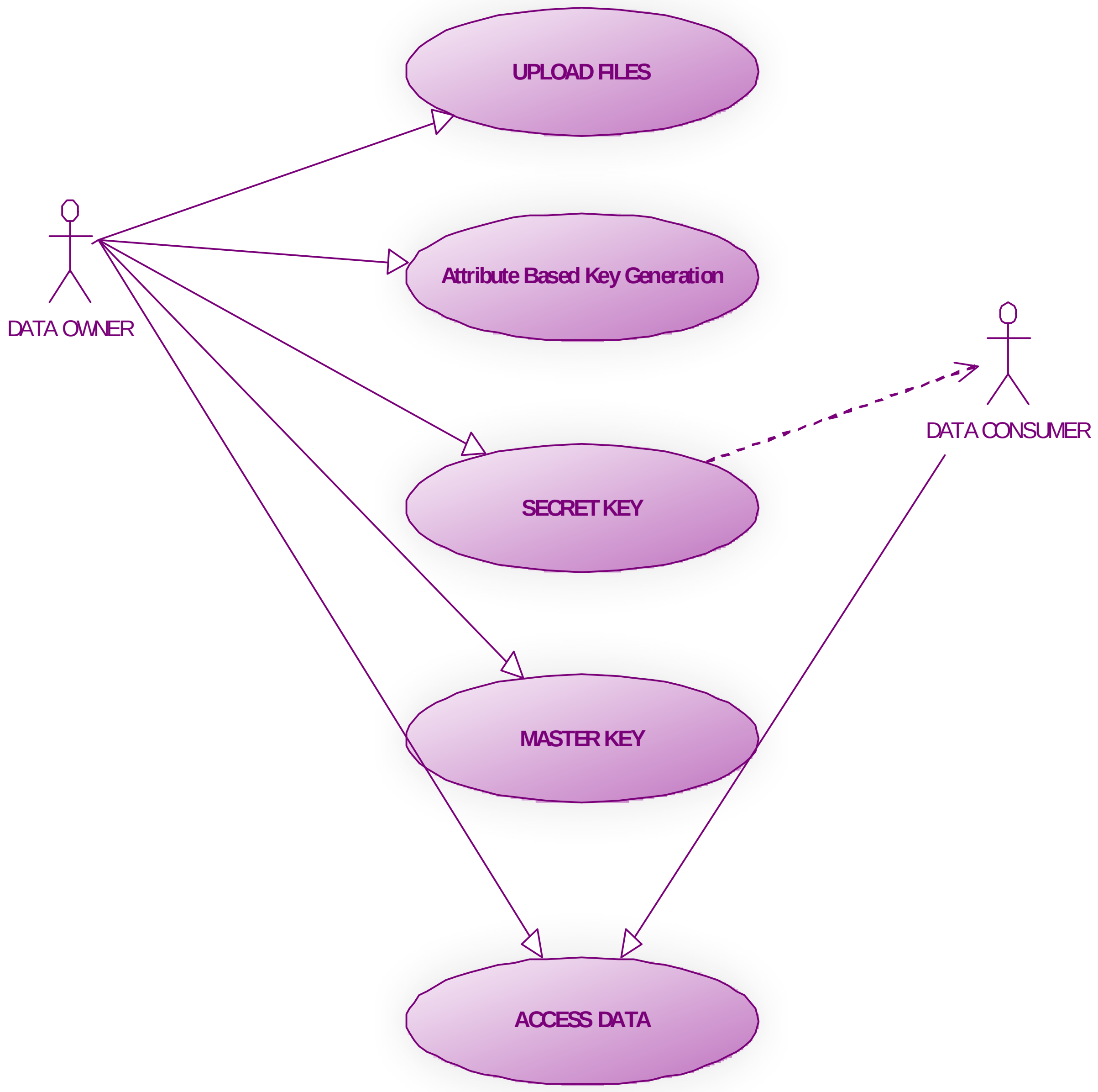
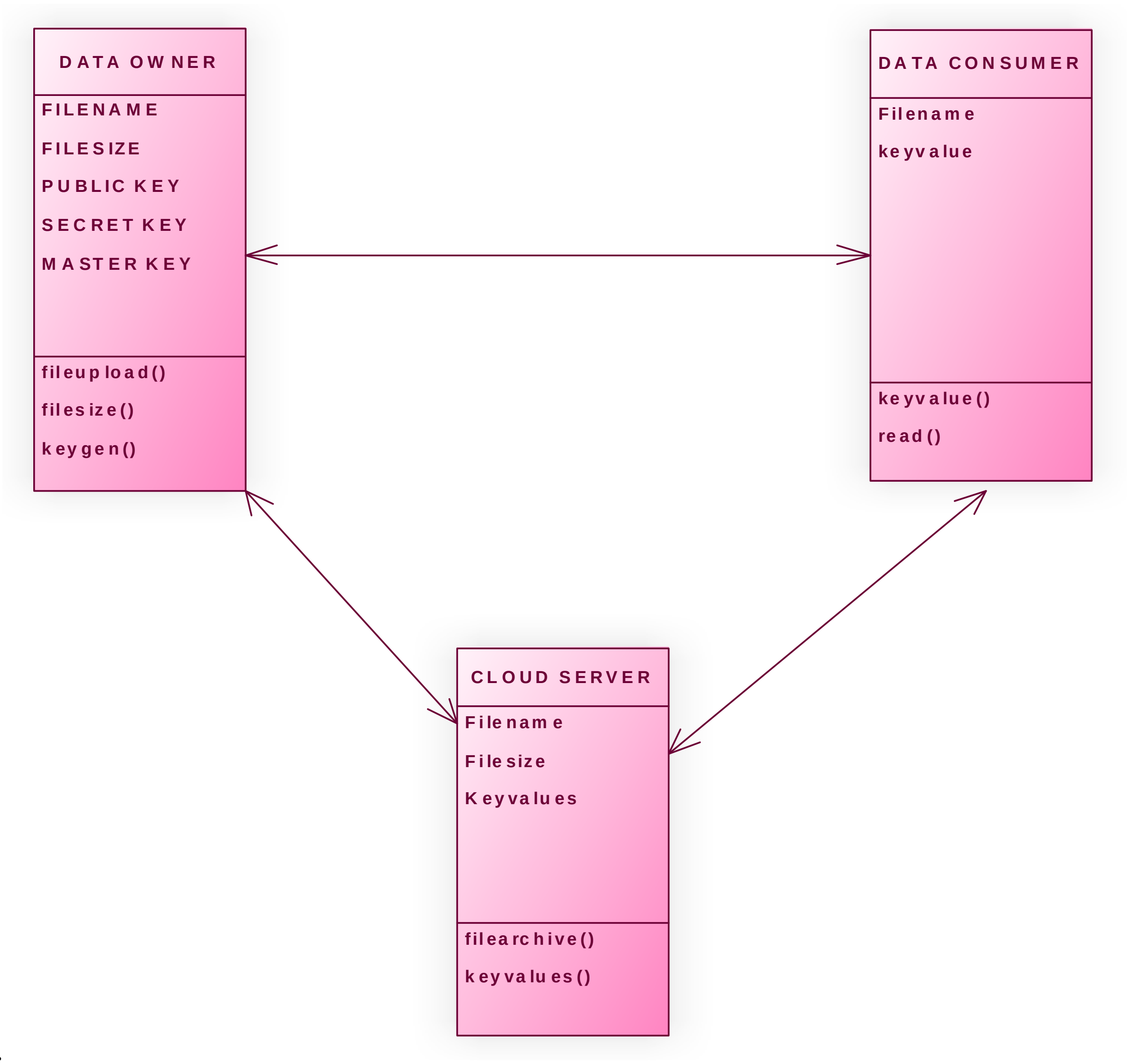


Fig. 6.6 Use-Case Diagram

6.5 CLASS DIAGRAM

A class diagram in the UML is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, and the relationships between the classes. Private visibility hides information from anything outside the class partition.



.

Fig. 6.7 Class Diagram

6.6 SYSTEM ARCHITECTURE

The nodes involved are admin and clients which stands as UI for the system. The deployment is performed as per the requirements of Hardware and software specified in the requirements phase.

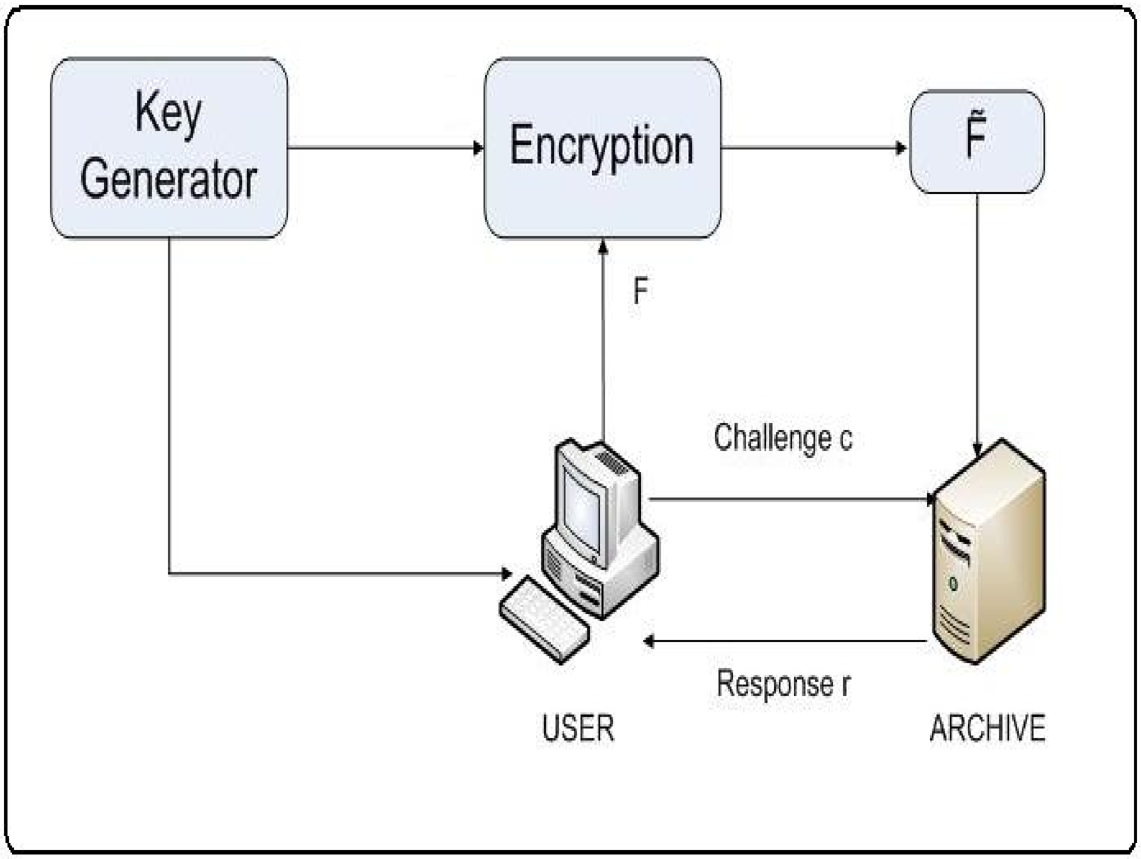


Fig. 6.8 System Architecture

CHAPTER 7

INPUT/OUTPUT DESIGN

7.1 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:

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.

7.2 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.
4. 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.

CHAPTER 8

SOFTWARE ENVIRONMENT

8.1 Java Technology

Java technology is both a programming language and a platform.

8.2 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.

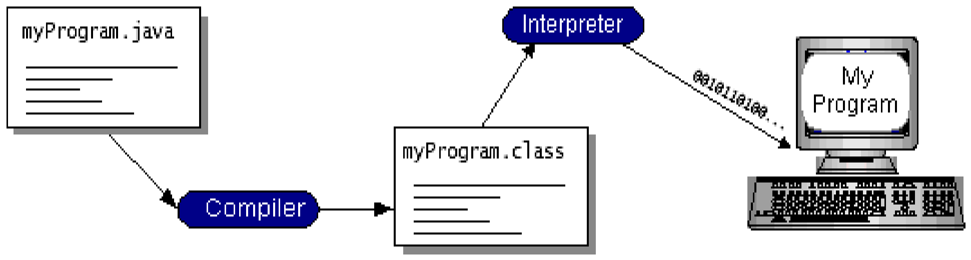


Fig 8.1: JAVA Program Execution Process

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.

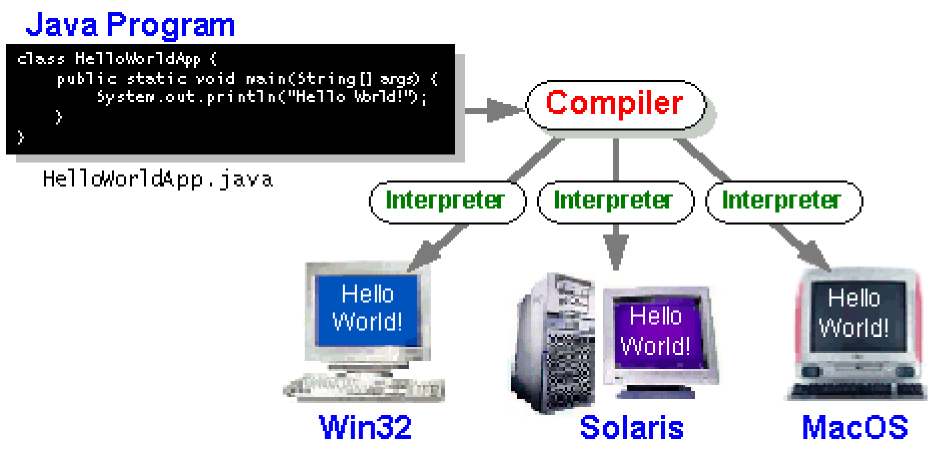

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Fig 8.2: JAVA virtual machine

8.3 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 Machos. 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 *p*ackages. 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.

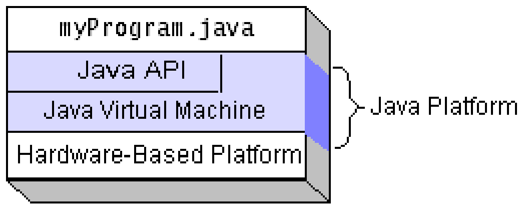


Fig 8.3: JAVA Platform

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.

8.4 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 JavaBeans^{TM}, can plug into existing component architectures.
- **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC^{TM})**: 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.

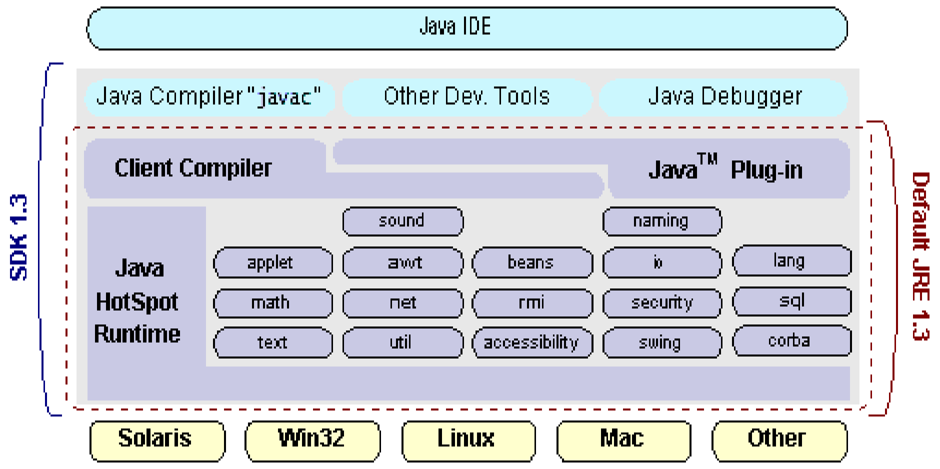


Fig 8.4: JAVA2 SDK 1.3

8.5 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 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.

8.6 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.

8.7 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.

8.8 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:

8.9 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.

8.9.1 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.

3. **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.

4. **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.

5. **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 precede the implementation using Java Networking. And for dynamically updating the cache table we go for MS Access database. Java has two things: a programming language and a platform.

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

- Simple
- Object-oriented
- Distributed
- Interpreted
- Robust
- 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.

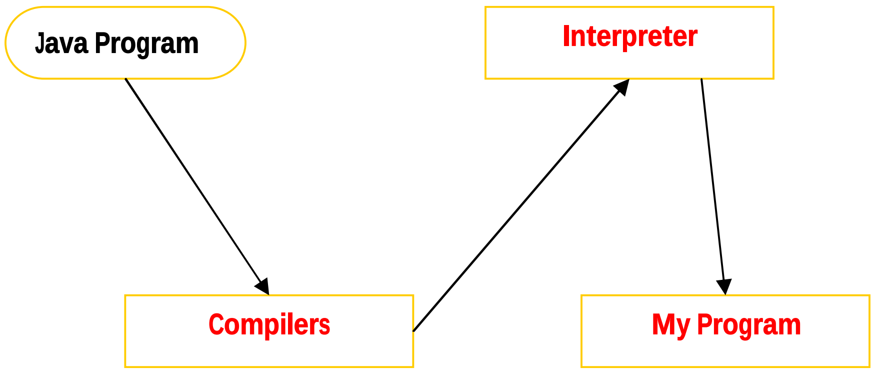


Fig 8.5: JAVA Compilation & Interpretation

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.

8.10 NETWORKING

8.10.1 TCP/IP stack

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

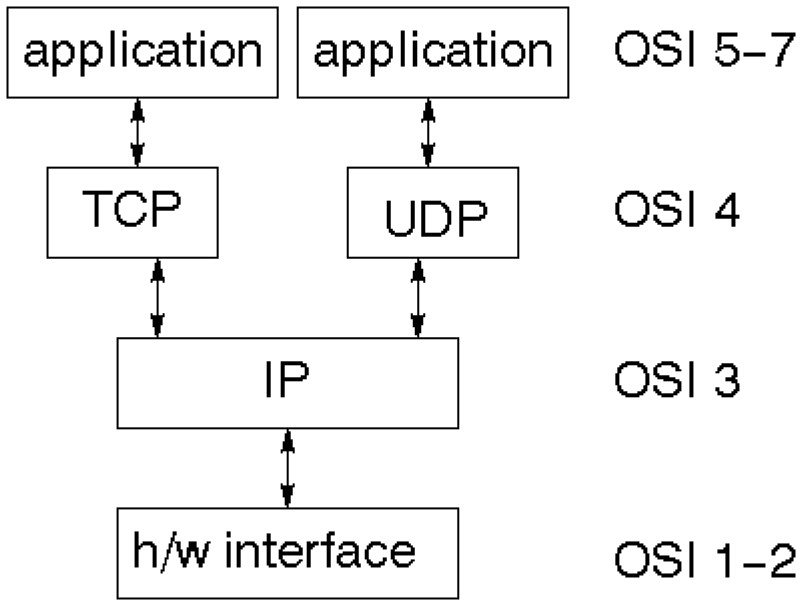


Fig 8.6: TCP/IP Stack

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

8.10.2 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.

8.10.3 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.

8.10.4 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.

8.10.5 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.

8.10.6 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.

8.10.7 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.

8.10.8 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.

8.10.9 Total address

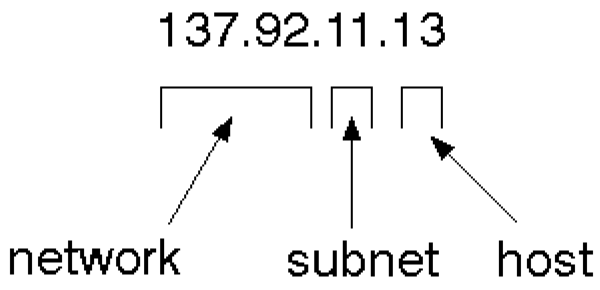


Fig 8.7: IP address Format

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

8.10.10 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".

8.10.11 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.

8.11 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. It is distributed under the terms of the GNU Lesser General Public Licence (LGPL), which permits use in proprietary applications.

8.11.1 Map Visualizatiztion

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.

8.11.2 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.

8.11.3 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.

8.11.4 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.

CHAPTER 9

IMPLEMENTATION

9.1 INTRODUCTION

This chapter describes the implementation of HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing. It deals with the source code for HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing.

9.2 SOURCE CODE

9.2.1 CONSUMER.JAVA

```
package com.action;
import com.util.DbConnector;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class Consumer extends HttpServlet {
 /**
 * Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
 * @param request servlet request
 * @param response servlet response
* @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
 protected void processRequest(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException {
 response.setContentType("text/html;charset=UTF-8");
 PreparedStatement pst = null;
 Connection conn = null;
 try{
 conn=(Connection) DbConnector.getConnection();
 String name= request.getParameter("_name").trim();
 String password= request.getParameter("_password").trim();
 String age =request.getParameter("_age").trim();
String emailid= request.getParameter("_emailid").trim();
 String phone= request.getParameter("_phone").trim();
 String sex= request.getParameter("add");
 String role= request.getParameter("role");
if( name != null && name!="" && password != null && password!=""  && age != null && age!="" && emailid != null && emailid!="" && phone!="" && phone != null && sex!="" && sex != null){
String sql="insert into user values ('"+name+"','"+password+"','"+sex+"','"+age+"','"+emailid+"',now(),'"+phone+"','consumer')";
System.out.println(">>"+sql);
 pst = (PreparedStatement) conn.prepareStatement(sql);
int a=pst.executeUpdate();
if(a >0){
response.sendRedirect("consumer.jsp?msg=Registration Done");
} else {
response.sendRedirect("consumer.jsp?msg=UserId Already Exit");
}
}
else{
```

```
response.sendRedirect("domAuthReg.jsp?msg=Fill All The Fileds");
} }catch(Exception e){
response.sendRedirect("domAuthReg.jsp?msg=UserId Already Exit");
e.printStackTrace();
}
}
// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the
left to edit the code.">
/**
* Handles the HTTP <code>GET</code> method.
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
 @Override
 protected void doGet(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException {
 processRequest(request, response);
}
 /**
* Handles the HTTP <code>POST</code> method.
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
 @Override
 protected void doPost(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException {
```

```
 processRequest(request, response);
 }
 /**
* Returns a short description of the servlet.
* @return a String containing servlet description
*/
 @Override
public String getServletInfo() {
return "Short description";
}// </editor-fold>
}
```

9.2.2 CONSUMERLOGIN.JAVA

```
package com.action;
import com.util.LoginProcessor;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
public class ConsumerLogin extends HttpServlet {
/**
* Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
* @param request servlet request
* @param response servlet response

* @throws ServletException if a servlet-specific error occurs

* @throws IOException if an I/O error occurs
```

```
*/
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
try {
if(LoginProcessor.getUserDetailsClient(request.getParameter("name_"),
request.getParameter("password"), "consumer")) {
HttpSession session = request.getSession();
session.setAttribute("userid", request.getParameter("name_"));
System.out.println(">>>>>>>" + session.getAttribute("userid"));
response.sendRedirect("consumerHome.jsp");
} else {
 response.sendRedirect("consumerLogin.jsp?msg=Check Userid or Password");
}
} catch (Exception e) {
 e.printStackTrace();
 }
}
 // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on
the left to edit the code.">
 /**
* Handles the HTTP <code>GET</code> method.

* @param request servlet request

* @param response servlet response

* @throws ServletException if a servlet-specific error occurs

* @throws IOException if an I/O error occurs

*/
```

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
 /**
* Handles the HTTP <code>POST</code> method.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
 /**
* Returns a short description of the servlet.
* @return a String containing servlet description
*/
@Override
public String getServletInfo() {
return "Short description";
}// </editor-fold>
}
```

9.2.3 DOMAINLOGIN.JAVA

```
package com.action;
import com.util.LoginProcessor;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
public class DomainLogin extends HttpServlet {
 /**
 * Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
try{
if(LoginProcessor.getUserDetailsClient(request.getParameter("name_"),
request.getParameter("password") ,"domain")) {
HttpSession session=request.getSession();

session.setAttribute("userid",request.getParameter("name_") );
System.out.println(">>>>>>>"+session.getAttribute("userid"));
response.sendRedirect("domAuthHome.jsp");
} else {
response.sendRedirect("domainLogin.jsp?msg=Check Userid or Password");
}
}catch(Exception e){
```

```
e.printStackTrace();
}
}
// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the
left to edit the code.">
/**
* Handles the HTTP <code>GET</code> method.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
@Override
 protected void doGet(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException {
 processRequest(request, response);
}
/**
* Handles the HTTP <code>POST</code> method.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
@Override
 protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
/**
```

```
* Returns a short description of the servlet.
* @return a String containing servlet description
*/
@Override
public String getServletInfo() {
return "Short description";
}// </editor-fold>
}
```

9.2.4 DBCONNECT.JAVA

```
package com.util;
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
import java.sql.*;
public class DbConnector {
// System.out.println("MySQL Connect Example.");
public static Connection getConnection() {
Connection conn = null;
String url = "jdbc:mysql://localhost:3306/";
String dbName = "hasbe";
String driver = "com.mysql.jdbc.Driver";
String userName = "root";
String password = "root";
try {
Class.forName(driver).newInstance();
conn = DriverManager.getConnection(url + dbName, userName, password);
System.out.println("Connected to the database");
//conn.close();
//System.out.println("Disconnected from database");
} catch (Exception e) {
e.printStackTrace();
}
return conn;
}
public static void main(String arg[]){
DbConnector.getConnection();
}
}
```

9.2.5 DATAOWNERHOME.JSP

```
<!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>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<meta name="description" content="" />
<meta name="keywords" content="" />
<title>HASBE</title>
<!--<linkhref="http://fonts.googleapis.com/css?family=Open+Sans"rel="stylesheet"
type="text/css" />-->
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<div id="wrapper">
<div id="header">
<div id="logo">
 <h1><a href="#">HASBE</a></h1>
```

```
</div>
<div id="menu">
<ul>
<li class="first current_page_item"><a href="#">Welcome Data Owner</a></li>
<li><a href="index.jsp">Log Out</a></li>
<br class="clearfix" />
</ul>
</div>
</div>
<div id="inner">
<div id="splash">
<%if (request.getParameter("msg") != null) {
out.println(request.getParameter("msg"));
}%>
<h2>Upload File</h2>
<form action="uploadProcess.jsp" method="post" enctype="multipart/form-data">
<tr style="height: 40px;"><td>Choose File:</td><td> <input name="file" type="file"
id="file"/>    </td></tr>
<tr style="height: 40px;"><td colspan="2"> <input type="submit" value="Upload"/>
</td></tr>
<table>
</table>
</form>
</div>
 </div>
</div>
</body>
</html>
```

9.2.6 DOMAINAUTHOME.JSP

```
<!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>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<meta name="description" content="" />
<meta name="keywords" content="" />
<title>HASBE</title>
<!--	<link	href="http://fonts.googleapis.com/css?family=Open+Sans"
rel="stylesheet" type="text/css" />-->
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<div id="wrapper">
<div id="header">
div id="logo">
<h1><a href="#">HASBE</a></h1>
</div>
<div id="menu">
<ul>
<li class="first current_page_item"><a href="#">Welcome Domain Authority! </a></li>
<li class="first current_page_item"><a href="domAuthHome.jsp">Data Owner</a></li>
<li><a href="consumer.jsp">Data Consumer</a></li>
<li><a href="index.jsp">LogOut</a></li>
</ul>
<br class="clearfix" />
</div>
</div>
```

```
<div id="inner">
<div id="splash">
<span style="font-size: 22px;">Data Owner Registration:</span>
<%if (request.getParameter("msg") != null) {
out.println(request.getParameter("msg"));
}%>
<form method="post" action="DataOwnerReg">
<table width="90%">
<tr style="height: 40px;"><td>User id:</td><td><input type="text" size="30" style="height: 25px;" id="_name" name="_name"/></td></tr>
<tr style="height: 40px;"><td>Password:</td><td><input type="password" size="30" style="height: 25px;" id="_password" name="_password"/></td></tr>
<tr style="height: 40px;"><td>Address:</td><td><input type="text" size="30" style="height: 25px;" id="add" name="add" />
</td></tr>
<tr style="height: 40px;"><td>Age:</td><td><input type="text" size="30" style="height: 25px;" id="_age" name="_age"/></td></tr>
<tr style="height: 40px;"><td>Phone:</td><td><input type="text" size="30" style="height: 25px;" id="_phone" name="_phone"/></td></tr>
<tr style="height: 40px;"><td>Email-Id:</td><td><input type="text" size="30" style="height: 25px;" id="_emailid" name="_emailid"/></td></tr>
<tr style="height: 40px;"><td></td><td><input type="submit" value="Submit" style="height: 25px;width: 90px;"/></td></tr>
</table>
</form>
</div>
</div>
</div>
</body>
</html>
```

9.2.7 WEB.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app                    version="3.0"                    xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
<servlet>
<servlet-name>DomAuthReg</servlet-name>
<servlet-class>com.action.DomAuthReg</servlet-class>
</servlet>
<servlet>
<servlet-name>UserReg</servlet-name>
<servlet-class>com.action.UserReg</servlet-class>
</servlet>
<servlet>
<servlet-name>UserLoginAction</servlet-name>
<servlet-class>com.action.UserLoginAction</servlet-class>
</servlet>
<servlet>
<servlet-name>DomainLogin</servlet-name>
<servlet-class>com.action.DomainLogin</servlet-class>
</servlet>
<servlet>
<servlet-name>TrustedLogin</servlet-name>
<servlet-class>com.action.TrustedLogin</servlet-class>
</servlet>
<servlet>
<servlet-name>DataOwnerReg</servlet-name>
<servlet-class>com.action.DataOwnerReg</servlet-class>
```

```
</servlet>
<servlet>
<servlet-name>Consumer</servlet-name>
<servlet-class>com.action.Consumer</servlet-class>
</servlet>
<servlet>
<servlet-name>OwnerLogin</servlet-name>
<servlet-class>com.action.OwnerLogin</servlet-class>
</servlet>
<servlet>
<servlet-name>ConsumerLogin</servlet-name>
<servlet-class>com.action.ConsumerLogin</servlet-class>
</servlet>
<servlet>
<servlet-name>UploadData</servlet-name>
<servlet-class>com.action.UploadData</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>DomAuthReg</servlet-name>
<url-pattern>/DomAuthReg</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>UserReg</servlet-name>
<url-pattern>/UserReg</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>UserLoginAction</servlet-name>
<url-pattern>/UserLoginAction</url-pattern>
</servlet-mapping>
<servlet-mapping>
```

```
<servlet-name>DomainLogin</servlet-name>
<url-pattern>/DomainLogin</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>TrustedLogin</servlet-name>
<url-pattern>/TrustedLogin</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>DataOwnerReg</servlet-name>
<url-pattern>/DataOwnerReg</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>Consumer</servlet-name>
<url-pattern>/Consumer</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>OwnerLogin</servlet-name>
<url-pattern>/OwnerLogin</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>ConsumerLogin</servlet-name>
<url-pattern>/ConsumerLogin</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>UploadData</servlet-name>
<url-pattern>/UploadData</url-pattern>
</servlet-mapping>
<session-config>
<session-timeout>
 30
```

```
</session-timeout>
</session-config>
<welcome-file-list>
<welcome-file>index.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

CHAPTER 10

SNAPSHOTS

10.1 INTRODUCTION

Snapshot is nothing but every moment of the application while running. It gives the clear elaborated of application. It will be useful for the new user to understand for the future steps.

10.2 HOME PAGE

Purpose: This is the home page form for authorization purpose.

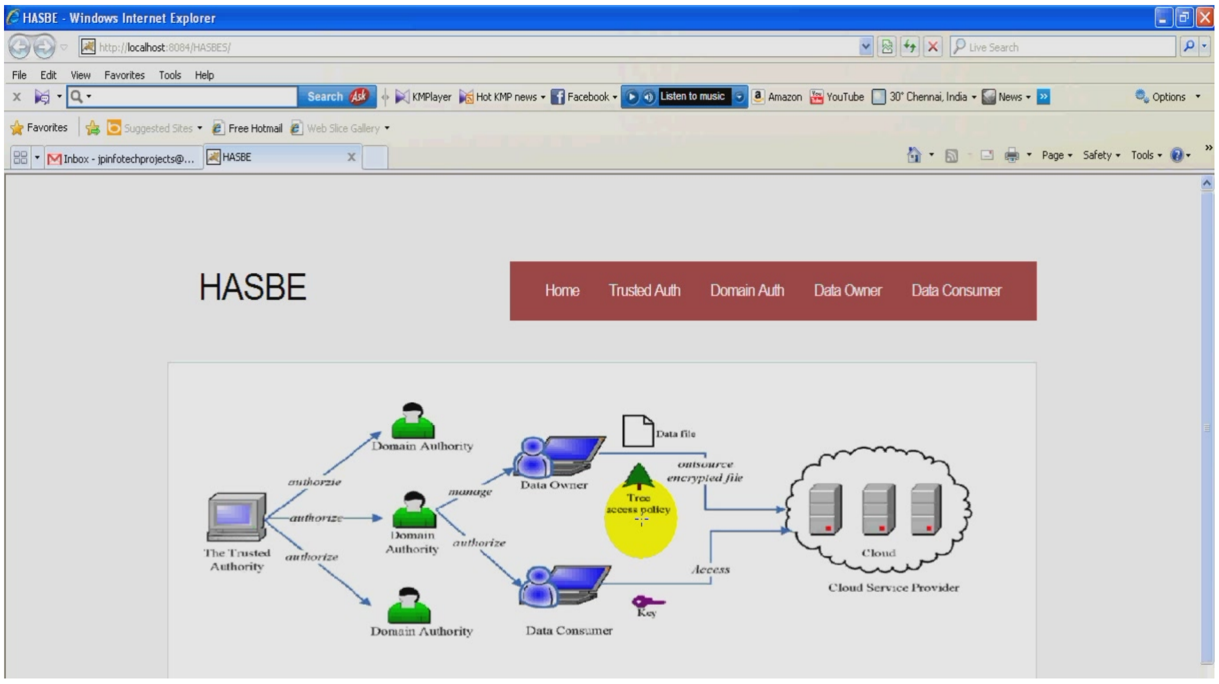


Fig. 10.1 Home Page

10.3 LOGIN PAGE

Purpose: This is the user login form for authorization purpose. If the user enters wrong username or password, the access will be denied.

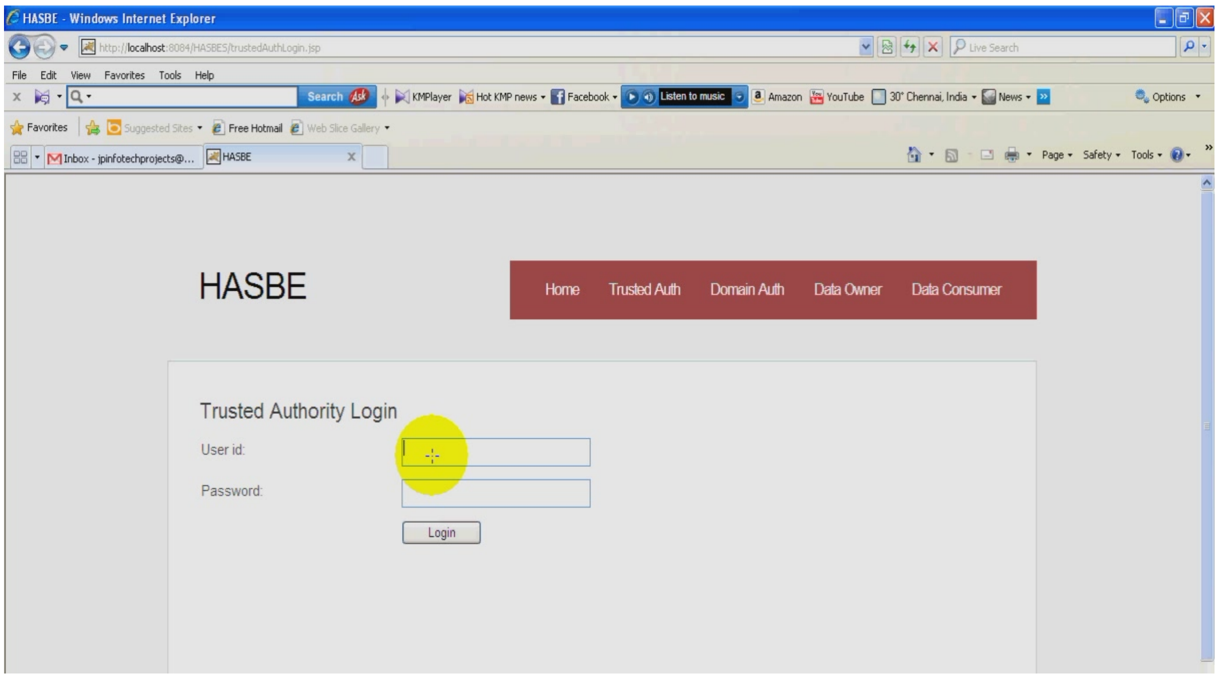
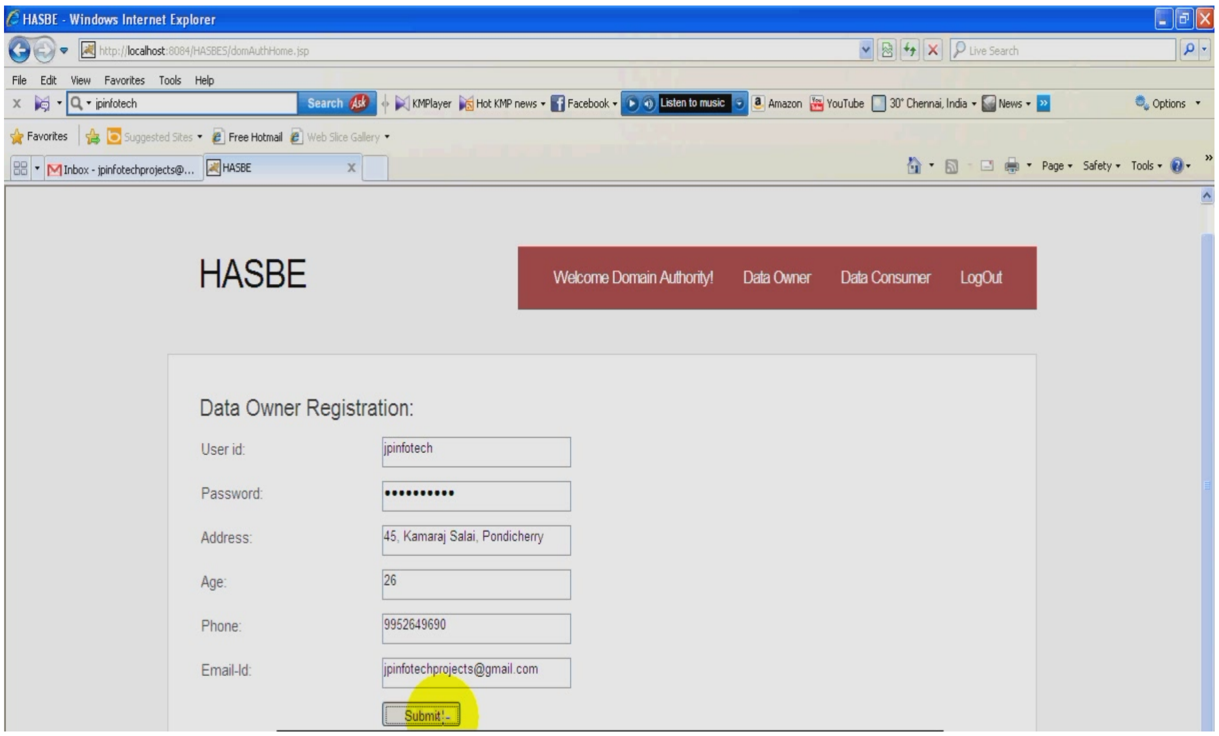


Fig. 10.2 Login page

10.4 REGISTRATION PAGE

PURPOSE: Data owner can create account by registering.


HASBE - Windows Internet Explorer
http://localhost:8084/HASBES/domAuthHome.jsp
HASBE
Welcome Domain Authority!
Data Owner
Data Consumer
LogOut
Submit

Data Owner Registration:

User id:	jpinfotech
Password:	••••••••••
Address:	45, Kamaraj Salai, Pondicherry
Age:	26
Phone:	9952649690
Email-Id:	jpinfotechprojects@gmail.com

Fig. 10.3 Registration Page

10.5 FILE UPLOAD

Purpose: Data owner can upload their data to the cloud server.

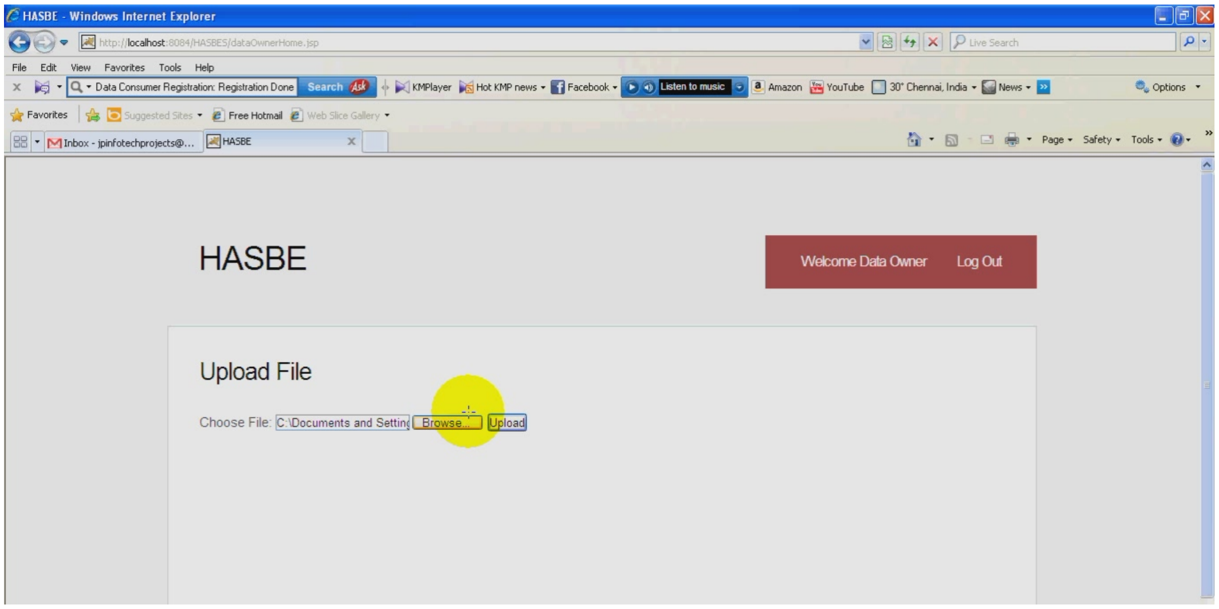
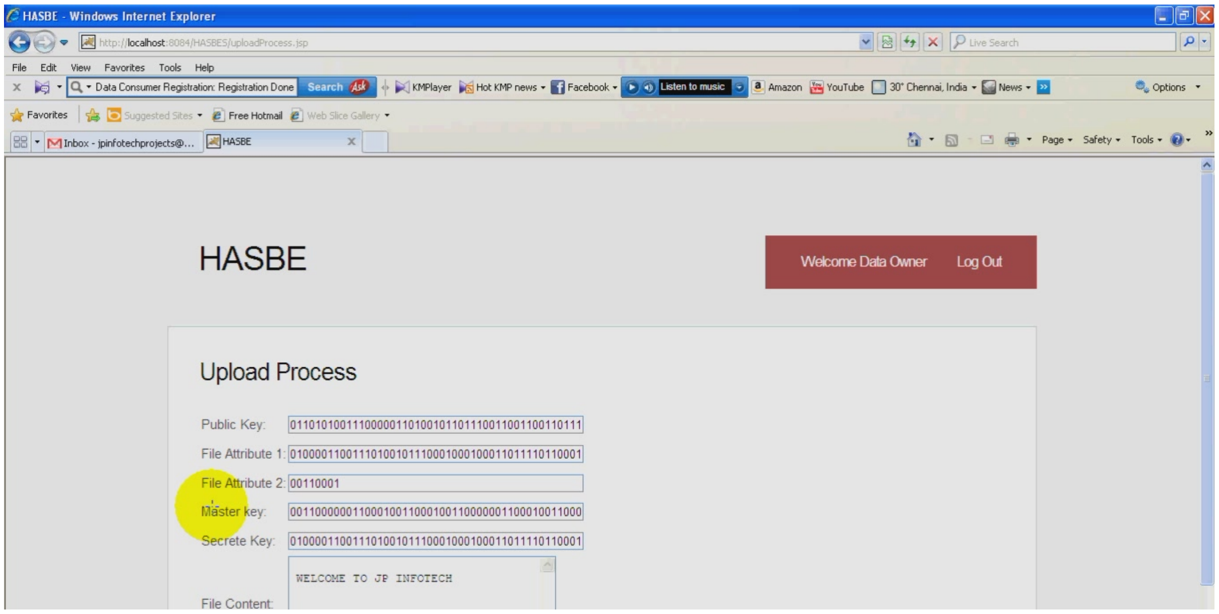


Fig 10.4 File Upload

10.6 UPLOAD PROCESS

Purpose: Data is uploading to the cloud server and generating secret key.


HASBE - Windows Internet Explorer
http://localhost:8084/HASBES/uploadProcess.jsp
HASBE
Welcome Data Owner
Log Out

Upload Process

Public Key:	0110101001110000011010010110111001100110011011 1
File Attribute 1:	0100001100111010010111000100010001101111011000 1
File Attribute 2:	00110001
Master key:	0011000000110001001100010011000000110001001100 0
Secrete Key:	0100001100111010010111000100010001101111011000 1
File Content:	WELCOME TO JP INFOTECH

Fig 10.5 Upload Process

10.7 CONSUMER LOGIN

Purpose: Consumer can login to access the data file from cloud server.

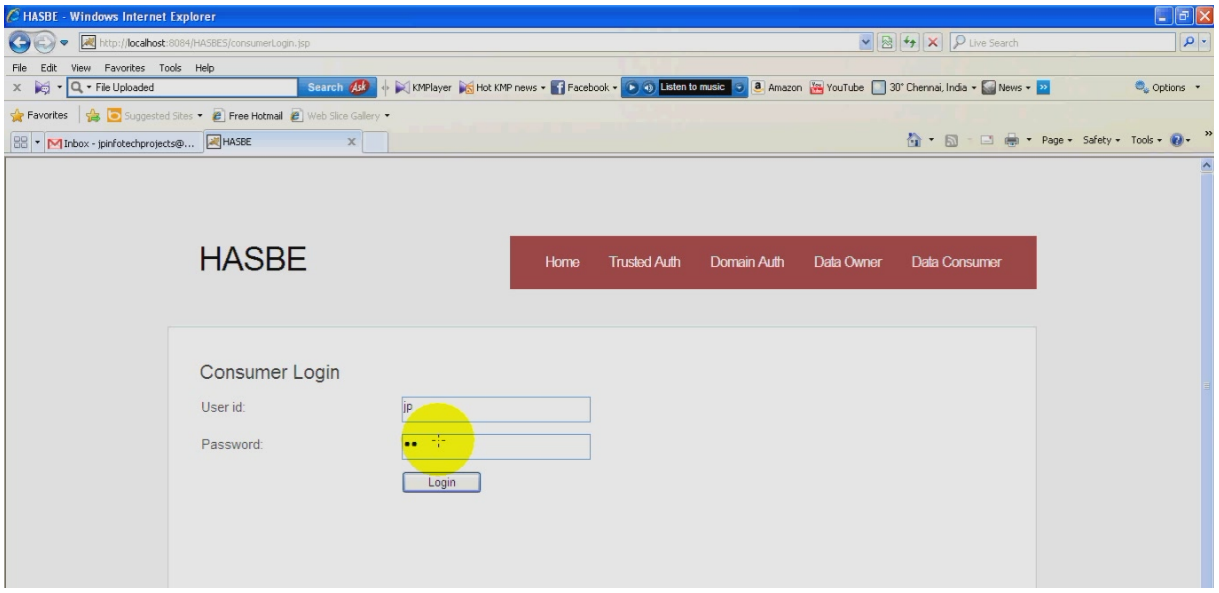
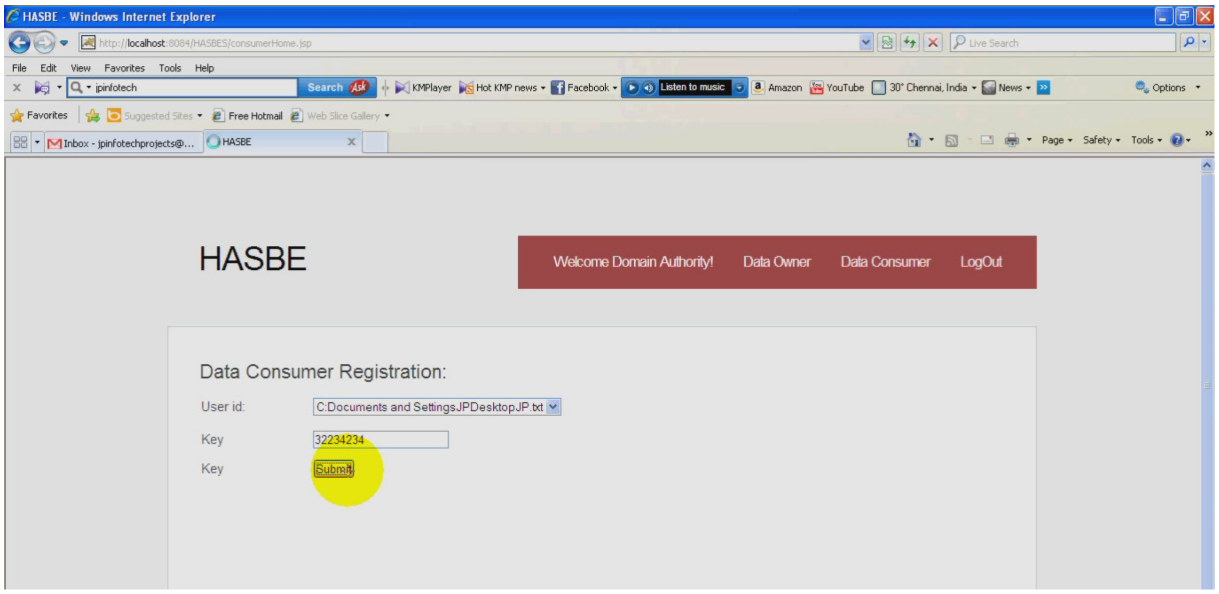


Fig 10.6 Consumer Login

10.8 DATA DOWNLOAD

Purpose: Consumer is accessing the data by entering the secret key.


HASBE - Windows Internet Explorer
http://localhost:8084/HASBES/consumerHome.jsp
HASBE
Welcome Domain Authority!
Data Owner
Data Consumer
LogOut
Submit

Data Consumer Registration:

User id: C:Documents and SettingsJPDesktopJP.txt

Key 32234234

Fig. 10.7 Data Download

CHAPTER 11

SOFTWARE TESTING

11.1 INTRODUCTION

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.

11.2 DEVELOPING METHODOLOGIES

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.

The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

11.3 Types of Testing

11.3.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces 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.

11.3.2 Functional testing

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.

11.3.3 System Testing

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.

11.3.4 Performance Testing

The Performance test ensures that the output is produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

11.3.5 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.

11.3.6 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.

Acceptance testing for Data Synchronization:

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node.
- The Route add operation is done only when there is a Route request in need.
- The Status of Nodes information is done automatically in the Cache Updating process.

11.3.7 Build the testing plan

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

CHAPTER 12

FUTURE ASPECTS

The project “HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing” have a greater future in the upcoming period. The use of Cloud Service is now increasing regularly. Security is a major issue in cloud as the use is increasing. This project provides security on various levels according to user type. We provide security to all the files by blocking the download option for all file. We can block each and every file separately for each and every user.

This project has also a greater future in the way of extension because of the software engineering approach is efficiently used during the development of the project. It can be used and implemented in real life.

CHAPTER 13

CONCLUSION

In this paper, we introduced the HASBE scheme for realizing scalable, flexible, and fine-grained access control in cloud computing. The HASBE scheme seamlessly incorporates a hierarchical structure of system users by applying a delegation algorithm to ASBE. HASBE not only supports compound attributes due to flexible attribute set combinations, but also achieves efficient user revocation because of multiple value assignments of attributes. We formally proved the security of HASBE based on the security of CP-ABE by Bethencourt et al... Finally, we implemented the proposed scheme, and conducted comprehensive performance analysis and evaluation, which showed its efficiency and advantages over existing schemes.

REFRENCES

[1] R. Buyya, C. ShinYeo, J. Broberg, and I. Brandic, “Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility,” Future Generation Comput. Syst., vol. 25, pp. 599–616, 2009.

[2] Amazon Elastic Compute Cloud (Amazon EC2) [Online]. Available: http://aws.amazon.com/ec2/

[3] Amazon Web Services (AWS) [Online]. Available: https://s3.amazonaws. com/

[4] R. Martin, “IBM brings cloud computing to earth with massive new data centers,” InformationWeek Aug. 2008 [Online]. Available: http:// www.informationweek.com/news/hardware/data_centers/209901523

[5] Google App Engine [Online]. Available: http://code.google.com/appengine/

[6] K. Barlow and J. Lane, “Like technology from an advanced alien culture: Google apps for education at ASU,” in *Proc. ACM* SIGUCCSUser Services Conf., Orlando, FL, 2007.

[7] B. Barbara, “Salesforce.com: Raising the level of networking,” Inf.Today, vol. 27, pp. 45–45, 2010.

[8] J. Bell, Hosting EnterpriseData in the Cloud—Part 9: InvestmentValue Zetta, Tech. Rep., 2010.

[9] A. Ross, “Technical perspective: A chilly sense of security,” Commun.*ACM*, vol. 52, pp. 90–90, 2009.

[10] D. E. Bell and L. J. LaPadula, Secure Computer Systems: Unified Exposition and Multics Interpretation The MITRE Corporation, Tech. Rep., 1976.

[11] K. J. Biba, Integrity Considerations for Secure Computer Sytems The MITRE Corporation, Tech. Rep., 1977.

[12] H. Harney, A. Colgrove, and P. D. McDaniel, “Principles of policy in secure groups,” in *Proc. NDSS*, San Diego, CA, 2001.

[13] P. D. McDaniel and A. Prakash, “Methods and limitations of security policy reconciliation,” in Proc. IEEE Symp. Security and Privacy, Berkeley, CA, 2002.

[14] T. Yu and M. Winslett, “A unified scheme for resource protection in automated trust negotiation,” in Proc. IEEE Symp. Security and Privacy, Berkeley, CA, 2003.

[15] J. Li, N. Li, and W. H. Winsborough, “Automated trust negotiation using cryptographic credentials,” in Proc. ACM Conf. Computer and Communications Security (CCS*)*, Alexandria, VA, 2005.

[16] V. Goyal, O. Pandey, A. Sahai, and B.Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in Proc. ACM Conf. Computer and Communications Security (ACM CCS), Alexandria, VA, 2006.

[17] S. Yu, C. Wang, K. Ren, and W. Lou, “Achieving secure, scalable, and fine-grained data access control in cloud computing,” in Proc. IEEE INFOCOM *2010*, 2010, pp. 534–542.

[18] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute based encryption,” in Proc. IEEE Symp. Security and Privacy, Oakland, CA, 2007.

[19] R. Bobba, H. Khurana, and M. Prabhakaran, “Attribute-sets: A practically motivated enhancement to attribute-based encryption,” in Proc. *ESORICS*, Saint Malo, France, 2009.

[20] A. Sahai and B. Waters, “Fuzzy identity based encryption,” in Proc. Acvances in Cryptology—Eurocrypt, 2005, vol. 3494, LNCS, pp. 457–473.

[21] G.Wang, Q. Liu, and J.Wu, “Hierachical attibute-based encryption for fine-grained access control in cloud storage services,” in Proc. ACM Conf. Computer and Communications Security (ACM CCS*)*, Chicago, IL, 2010.