A LIGHTWEIGHT SECURE DATA SHARING SCHEME FOR MOBILE CLOUD COMPUTING

Report submitted to

Sree Konaseema Bhanoji Ramars P.G College

For the award of the degree

Of

MASTER OF COMPUTER APPLICATIONS

By

ILLURI NAGA SURYA BHASKAR

Reg.No: 2081151018

Under the esteemed guidance of

Mr.M.S. VENU GOPALA RAO

M.Tech.,APSET.

Asst. Prof- Department of MCA

DEPARTMENT OF MCA

SREE KONASEEMA BHANOJI RAMARS POST GRADUATE COLLEGE

AMALAPURAM-533201

(Affiliated to Adikavi Nannaya University-Rajamahendravaram)

2022

DECLARATION

A. The work contained in this report is original and has been done by me under the guidance of my Supervisors Asst. Prof M.S. Venu Gopala Rao and Nagaraju Bonam, BSS Software Solutions.

B. The work has not been submitted to any other Institute for any degree or diploma.

C. I have followed the guidelines provided by the university in preparing the report.

D. Whenever I have used materials (data theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Signature of the Student

Sree Konaseema Bhanoji Ramars P.G College

(Affiliated to Adikavi Nannaya University)

CERTIFICATE

This is to certify that the project report entitled, **"A LIGHTWEIGHT SECURE DATA SHARING SCHEME FOR MOBILE CLOUD COMPUTING"** submitted by **Mr."Illuri Naga Surya Bhaskar" Regd No.2081151018**, to "**Sree Konaseema Bhanoji Ramars P.G College**", Affiliated to "**Adikavi Nannaya University**", Rajahmahendravaram , Andhra Pradesh, India, is a record of bonafide project work carried out by his under my/our supervision and guidance and is worthy of consideration for the award of the degree of Master of computer Aplications.

(Internal Examiner)	**(Head of the Department)**	**(External Examiner)**

Date:

ACKNOWLEDGEMENTS

I feel fortunate to pursue my **Master of Computer Applications** degree in the campus of **SREE KONASEEMA BHANGJI RAMARS P.G COLLEGE ,Amalapuram**. It provided all the facilities in the areas of MCA Software Lab.

It's a genuine pleasure to express my deep sense of thanks and gratitude to my mentor and project guidance **Nagaraju Bonam BSS SOFTWARE SOLUTIONS** right from selection of project and his valuable suggestions throughout the project work. His constant and timely counsel has been caused for to success, in completing this thesis in the college, he has given me a tremendous support in a technical front.

1 profusely thank **Dr. P. KRISHNA KISHORE, Principal of S.K.B.R P.G COLLEGE** for allthe encouragement and support

I also thank to **Asst. Prof. M. S. VENU GOPALA RAO, Head of the Department** MCA for the guidance throughout the academic.

A great deal of thanks goes to review committee members and the entire Faculty of Department of MCA S.K.B.R P.G COLLEGE, Amalapuram for their excellent supervision and valuable suggestions.

I am always thankful to my family, friends and well-wishers for all their trust and constant support in the successful completion of my project.

1. INTRODUCTION

1.1 Introduction:

with the development of cloud computing and the popularity of smart mobile devices, people are gradually getting accustomed to a new era of data sharing model in which the data is stored on the cloud and the mobile devices are used to store/retrieve the data from the cloud. Typically, mobile devices only have limited storage space and computing power. On the contrary, the cloud has enormous amount of resources. In such a scenario, to achieve the satisfactory performance, it is essential to use the resources provided by the cloud service provider (CSP) to store and share the data.

Nowadays, various cloud mobile applications have been widely used. In these applications, people (data owners) can upload their photos, videos, documents and other files to the cloud and share these data with other people (data users) they like to share. CSPs also provide data management functionality for data owners. Since personal data files are sensitive, data owners are allowed to choose whether to make their data files public or can only be shared with specific data users. Clearly, data privacy of the personal sensitive data is a big concern for many data owners.

The state-of-the-art privilege management/access control mechanisms provided by the CSP are either not sufficient or not very convenient. They cannot meet all the

Requirements of data owners. First, when people upload their data files onto the cloud, they are leaving the data in a place where is out of their control, and the CSP may spy on user data for its commercial interests and/or other reasons. Second, people have to send password to each data user if they only want to share the encrypted data with certain users, which is very cumbersome. To simplify the privilege management, the data owner can divide data users into different groups and send password to the groups which they want to share the data. However, this approach requires fine-grained access control. In both cases, password management is a big issue.

Apparently, to solve the above problems, personal sensitive data should be encrypted before uploaded onto the cloud so that the data is secure against the CSP. However, the data encryption brings new problems. How to provide

efficient access control mechanism on ciphertext decryption so that only the authorized users can access the plaintext data is challenging. In addition, system must offer data owners effective user privilege management capability, so they can grant/revoke data access privileges easily on the data users. There have been substantial researches on the issue of data access control over ciphertext. In these researches, they have the following common assumptions. First, the CSP is considered honest and curious. Second, all the sensitive data are encrypted before uploaded to the Cloud. Third, user authorization on certain data is achieved through encryption/decryption key distribution. In general, we can divide these approaches into four categories: simple ciphertext access control, hierarchical access control, access control based on fully homomorphic encryption [1][2] and access control based on attribute-based encryption (ABE). All these proposals are designed for non-mobile cloud environment. They consume large amount of storage and computation resources, which are not available for mobile devices. According to the experimental results in [26], the basic ABE operations take much longer time on mobile devices than laptop or desktop computers. It is at least 27 times longer to execute on a smart phone than a personal computer (PC). This means that an encryption operation which takes one minute on a PC will take about half an hour to finish on a mobile device. Furthermore, current solutions don't solve the user privilege change problem very well. Such an operation could result in very high revocation cost. This is not applicable for mobile devices as well. Clearly, there is no proper solution which can effectively solve the secure data sharing problem in mobile cloud. As the mobile cloud becomes more and more popular, providing an efficient secure data sharing mechanism in mobile cloud is in urgent need.

1.2 Purpose:

All these proposals are designed for non-mobile cloud environment. They consume large amount of storage and computation resources, which are not available for mobile devices. According to the experimental results in the basic ABE operations take much longer time on mobile devices than laptop or desktop computers. It is at least 27 times longer to execute on a smart phone than a personal computer (PC). This means that an encryption operation which takes one minute on a PC will take about half an hour to finish on a mobile device. Furthermore, current solutions don't solve the user privilege change problem very well. Such

an operation could result in very high revocation cost. This is not applicable for mobile devices as well. Clearly, there is no proper solution which can effectively solve the secure data sharing problem in mobile cloud. As the mobile cloud becomes more and more popular, providing an efficient secure data sharing mechanism in mobile cloud is in urgent need.

1.3 Scope:

LDSS is designed under the same assumptions proposed in 0 that the CSP is honest but curious, which means that the CSP will faithfully execute the operations requested by users, but it will peek on what users have stored in the cloud. The CSP will faithfully store users' data, undertake an initial access control, update data according to users' requests. However, CSP may do malicious actions such as collusion with users to get the data in plain text.

1.4 Motivation:

The state-of-the-art privilege management/access control mechanisms provided by the CSP are either not sufficient or not very convenient. They cannot meet all the requirements of data owners. First, when people upload their data files onto the cloud, they are leaving the data in a place where is out of their control, and the CSP may spy on user data for its commercial interests and/or other reasons. Second, people have to send password to each data user if they only want to share the encrypted data with certain users, which is very cumbersome. To simplify the privilege management, the data owner can divide data users into different groups and send password to the groups which they want to share the data. However, this approach requires fine-grained access control. In both cases, password management is a big issue.

1.5 Overview:

Attribute-based encryption (ABE) is proposed by Sahai and Waters [29]. It is derived from the Identity-Based Encryption (IBE) and is particularly suitable for one-tomany data sharing scenarios in a distributed and open cloud environment. Attribute-based encryption is divided into two categories: one is the Ciphertext-Policy Attribute Based Encryption (CP-ABE), in which the access control policy is embedded into ciphertext; the other one is KeyPolicy Attribute Based Encryption (KP-ABE), in which the access control policy is embedded in the user's key attributes. In real applications, CP-ABE is more suitable since it resembles role-based access control.

2. LITERATURE SURVEY

1) Attribute-based fine-grained access control with efficient revocation in cloud storage systems

AUTHORS: Kan Yang, Xiaohua Jia, Kui Ren

A cloud storage service allows data owner to outsource their data to the cloud and through which provide the data access to the users. Because the cloud server and the data owner are not in the same trust domain, the semi-trusted cloud server cannot be relied to enforce the access policy. To address this challenge, traditional methods usually require the data owner to encrypt the data and deliver decryption keys to authorized users. These methods, however, normally involve complicated key management and high overhead on data owner. In this paper, we design an access control framework for cloud storage systems that achieves fine-grained access control based on an adapted Ciphertext-Policy Attribute-based Encryption (CP-ABE) approach. In the proposed scheme, an efficient attribute revocation method is proposed to cope with the dynamic changes of users' access privileges in large-scale systems. The analysis shows that the proposed access control scheme is provably secure in the random oracle model and efficient to be applied into practice.

2) Achieving Usable and Privacy-assured Similarity Search over Outsourced Cloud Data

AUTHORS: Cong Wang, Kui Ren, Shucheng Yu, and Karthik Mahendra Raje

As the data produced by individuals and enterprises that need to be stored and utilized are rapidly increasing, data owners are motivated to outsource their local complex data management systems into the cloud for its great flexibility and economic savings. However, as sensitive cloud data may have to be encrypted before outsourcing, which obsoletes the traditional data utilization service based on plaintext keyword search, how to enable privacy-assured utilization mechanisms for outsourced cloud data is thus of paramount importance. Considering the large number of on-demand data users and huge amount of outsourced data files in cloud, the problem is particularly challenging, as it is extremely difficult to meet also the practical requirements of performance, system usability, and high-level user searching experiences. In this

paper, we investigate the problem of secure and efficient similarity search over outsourced cloud data. Similarity search is a fundamental and powerful tool widely used in plaintext information retrieval, but has not been quite explored in the encrypted data domain. Our mechanism design first exploits a suppressing technique to build storage-efficient similarity keyword set from a given document collection, with edit distance as the similarity metric. Based on that, we then build a private trie-traverse searching index, and show it correctly achieves the defined similarity search functionality with constant search time complexity. We formally prove the privacy-preserving guarantee of the proposed mechanism under rigorous security treatment. To demonstrate the generality of our mechanism and further enrich the application spectrum, we also show our new construction naturally supports fuzzy search, a previously studied notion aiming only to tolerate typos and representation inconsistencies in the user searching input. The extensive experiments on Amazon cloud platform with real data set further demonstrate the validity and practicality of the proposed mechanism.

3) **DACMACS: Effective Data Access Control for Multiauthority Cloud Storage Systems**

AUTHORS: Kan Yang, Xiaohua Jia, Kui Ren, Bo Zhang, Ruitao Xie

Data access control is an effective way to ensure data security in the cloud. However, due to data outsourcing and untrusted cloud servers, the data access control becomes a challenging issue in cloud storage systems. Existing access control schemes are no longer applicable to cloud storage systems, because they either produce multiple encrypted copies of the same data or require a fully trusted cloud server. Ciphertext-policy attribute-based encryption (CP-ABE) is a promising technique for access control of encrypted data. However, due to the inefficiency of decryption and revocation, existing CP-ABE schemes cannot be directly applied to construct a data access control scheme for multiauthority cloud storage systems, where users may hold attributes from multiple authorities. In this paper, we propose data access control for multiauthority cloud storage (DAC-MACS), an effective and secure data access control scheme with efficient decryption and revocation. Specifically, we construct a new multiauthority CP-ABE scheme with efficient decryption, and also design an efficient attribute revocation method that can achieve both forward security and

backward security. We further propose an extensive data access control scheme (EDAC-MACS), which is secure under weaker security assumptions.

4) Attribute based proxy re-encryption with delegating capabilities.

AUTHORS: Liang Xiaohui, Cao Zhenfu, Lin Huang

Attribute based proxy re-encryption scheme (ABPRE) is a new cryptographic primitive which extends the traditional proxy re-encryption (public key or identity based cryptosystem) to the attribute based counterpart, and thus empower users with delegating capability in the access control environment. Users, identified by attributes, could freely designate a proxy who can re-encrypt a ciphertext related with a certain access policy to another one with a different access policy. The proposed scheme is proved selective-structure chosen plaintext secure and master key secure without random oracles. Besides, we develop another kind of key delegating capability in our scheme and also discuss some related issues including a stronger security model and applications.

5) Attribute based proxy re-encryption with delegating capabilities

AUTHORS: Liang Xiaohui, Cao Zhenfu, Lin Huang

Attribute based proxy re-encryption scheme (ABPRE) is a new cryptographic primitive which extends the traditional proxy re-encryption (public key or identity based cryptosystem) to the attribute based counterpart, and thus empower users with delegating capability in the access control environment. Users, identified by attributes, could freely designate a proxy who can re-encrypt a ciphertext related with a certain access policy to another one with a different access policy. The proposed scheme is proved selective-structure chosen plaintext secure and master key secure without random oracles. Besides, we develop another kind of key delegating capability in our scheme and also discuss some related issues including a stronger security model and applications.

3. SYSTEM ANALYSIS

3.1 Introduction:

The first step in developing anything is to state the requirements. This applies just as much to leading edge research as to simple programs and to personal programs, as well as to large team efforts. Being vague about your objective only postpones decisions to a later stage where changes are much more costly.

The problem statement should state what is to be done and not how it is to be done. It should be a statement of needs, not a proposal for a solution. A user manual for the desired system is a good problem statement. The requestor should indicate which features are mandatory and which are optional, to avoid overly constraining design decisions. The requestor should avoid describing system internals, as this restricts implementation flexibility. Performance specifications and protocols for interaction with external systems are legitimate requirements. Software engineering standards, such as modular construction, design for testability, and provision for future extensions, are also proper.

Many problems statements from individuals companies and government agencies mixture requirements with design decisions. There may sometimes be a compelling reason to require a particular computer or language; there is rarely justification to specify the use of a particular algorithm. The analyst must separate the true requirements from design and implementation decisions disguised as requirements. The analyst should challenge such pseudo requirements, as they restrict flexibility. There may be politics or organizational reasons for the requirements, but at least the analyst should recognize that these externally imposed design decisions are not essential features of the problem domain.

A problem statement may have more or less detail. A requirement for a conventional product, such as a payroll program or a billing system, may have considerable detail. A requirement for a research effort in a new area may lack many details, but presumably the research has some objective, which should be clearly stated.

Most problem statements are ambiguous, incomplete, or even inconsistent. Some requirements are just plain wrong. Some requirements, although precisely stated, have unpleasant consequences on the system behavior or impose unreasonable implementation costs. Some requirements seem reasonable at first but do not work out

as well as the request or thought. The problem statement is just a starting point for understanding the problem, not an immutable document. The purpose of the subsequent analysis is to fully understand the problem and its implications. There is no reasons to expect that a problem statement prepared without a fully analysis will be correct.

The analyst must work with the requestor to refine the requirements so they represent the requestor's true intent. This involves challenging the requirements and probing for missing information. The psychological, organizational, and political considerations of doing this are beyond the scope of this book, except for the following piece of advice If you do exactly what the customer asked for, but the result does not meet the customer's real needs, you will probably be blamed anyway.

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies that people use to develop these systems.

In software engineering the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system the software development process.

Software Model OR Architecture analysis

Structured project management techniques (such as an SDLC) enhance management's control over projects by dividing complex tasks into manageable sections. A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. But none of the SDLC models discuss the key issues like Change management, Incident management and Release management processes within the SDLC process, but, it is addressed in the overall project management. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. The one size fits all approach to applying SDLC methodologies is no longer appropriate. We have made an attempt to

address the above mentioned defects by using a new hypothetical model for SDLC described elsewhere. The drawback of addressing these management processes under the overall project management is missing of key technical issues pertaining to software development process that is, these issues are talked in the project management at the surface level but not at the ground level.

What is SDLC

A software cycle deals with various parts and phases from planning to testing and deploying software. All these activities are carried out in different ways, as per the needs. Each way is known as a Software Development Lifecycle Model (SDLC).A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. A descriptive model describes the history of how a particular software system was developed. Descriptive models may be used as the basis for understanding and improving software development processes or for building empirically grounded prescriptive models.

SDLC models

- **The Linear model (Waterfall):** Separate and distinct phases of specification and development All activities in linear fashion Next phase starts only when first one is complete.
- **Evolutionary development:** Specification and development are interleaved (Spiral, incremental, prototype based, Rapid Application development) Incremental Model (Waterfall in iteration) RAD(Rapid Application Development) Focus is on developing quality product in less time,
- **Spiral Model:** We start from smaller module and keeps on building it like a spiral. It is also called Component based development.
- **Formal systems development:** A mathematical system model is formally transformed to an implementation.
- **Agile Methods:** Inducing flexibility into development.
- **Reuse-based development:** The system is assembled from existing components.

The General Model

Software life cycle models describe phases of the software cycle and the order in which those phases are executed. There are tons of models, and many companies adopt their own, but all have very similar patterns. The general, basic model is shown below:

General Life Cycle Model:

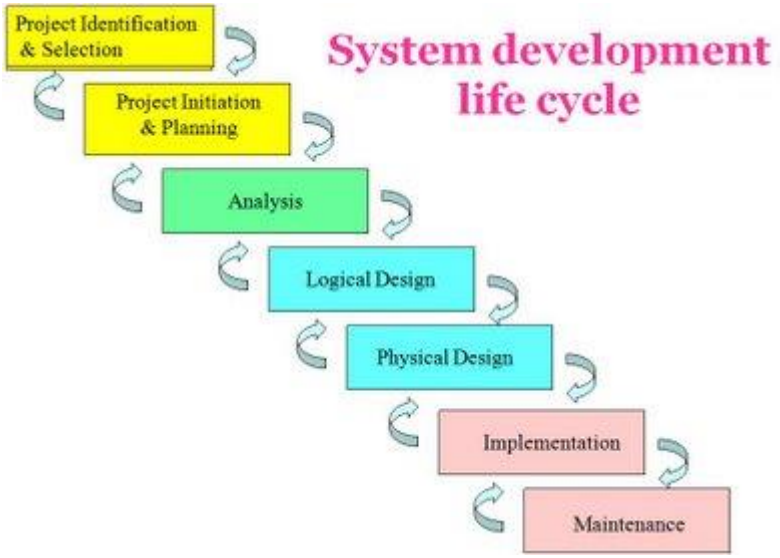


Fig 3.1 System development Life Cycle

Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced during implementation that is driven by the design. Testing verifies the deliverable of the implementation phase against requirements.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, "A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as

- The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system.
- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved by a fourfold procedure:
- Evaluating the first prototype in terms of its strengths, weakness, and risks.
- Defining the requirements of the second prototype.
- Planning an designing the second prototype.
- Constructing and testing the second prototype.
- At the customer option, the entire project can be aborted if the risk is deemed too great Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.
- The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
- The final system is constructed, based on the refined prototype.
- The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

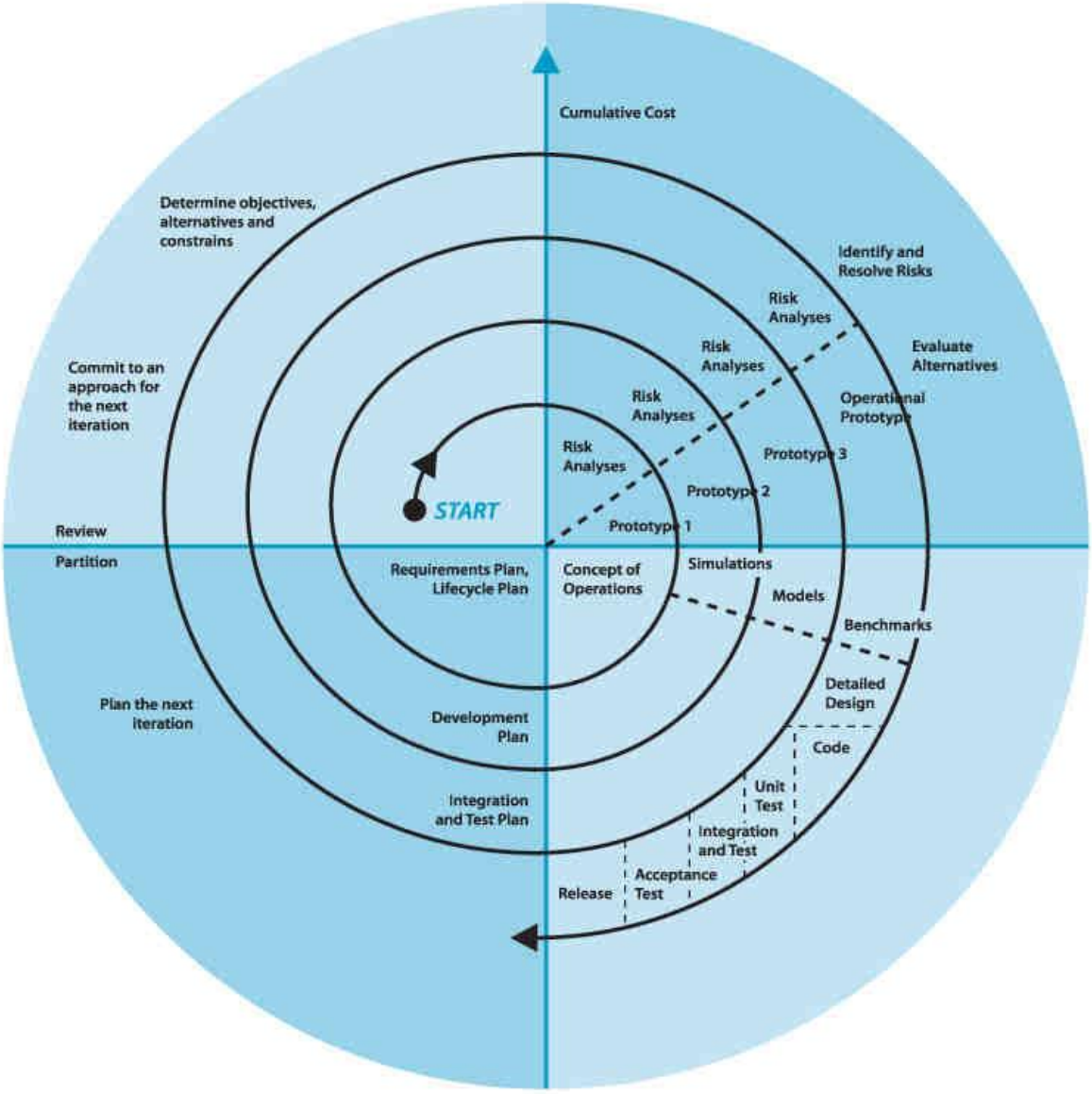


Fig3.2 Spiral Model

Advantages:

- Estimates(i.e. budget, schedule etc .) become more relistic as work progresses, because important issues discoved earlier.
- It is more able to cope with the changes that are software development generally entails.
- Software engineers can get their hands in and start woring on the core of a project earlier.

3.2 Existing system:

- In general, we can divide these approaches into four categories: simple ciphertext access control, hierarchical access control, access control based on fully homomorphic encryption and access control based on attribute-based encryption (ABE). All these proposals are designed for non-mobile cloud environment
- Tysowski et al. considered a specific cloud computing environment where data are accessed by resource-constrained mobile devices, and proposed novel modifications to ABE, which assigned the higher computational overhead of cryptographic operations to the cloud provider and lowered the total communication cost for the mobile user.

3.2.1 Disadvantages:

- Data privacy of the personal sensitive data is a big concern for many data owners.
- The state-of-the-art privilege management/access control mechanisms provided by the CSP are either not sufficient or not very convenient.
- They cannot meet all the requirements of data owners.
- They consume large amount of storage and computation resources, which are not available for mobile devices
- Current solutions don't solve the user privilege change problem very well. Such an operation could result in very high revocation cost. This is not applicable for mobile devices as well. Clearly, there is no proper solution which can effectively solve the secure data sharing problem in mobile cloud.

3.3 Proposed System:

- We propose a Lightweight Data Sharing Scheme (LDSS) for mobile cloud computing environment.
- The main contributions of LDSS are as follows:
- We design an algorithm called LDSS-CP-ABE based on Attribute-Based Encryption (ABE) method to offer efficient access control over ciphertext.
- We use proxy servers for encryption and decryption operations. In our approach, computational intensive operations in ABE are conducted on proxy servers, which greatly reduce the computational overhead on client side

mobile devices. Meanwhile, in LDSS-CP-ABE, in order to maintain data privacy, a version attribute is also added to the access structure. The decryption key format is modified so that it can be sent to the proxy servers in a secure way.

- We introduce lazy re-encryption and description field of attributes to reduce the revocation overhead when dealing with the user revocation problem.
- Finally, we implement a data sharing prototype framework based on LDSS.

3.3.1Advantages:

- The experiments show that LDSS can greatly reduce the overhead on the client side, which only introduces a minimal additional cost on the server side.
- Such an approach is beneficial to implement a realistic data sharing security scheme on mobile devices.
- The results also show that LDSS has better performance compared to the existing ABE based access control schemes over ciphertext.
- Multiple revocation operations are merged into one, reducing the overall overhead
- In LDSS, the storage overhead needed for access control is very small compared to data files.

3.4 Feasibility Study:

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

3.4.1 Economical Feasibility

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the

development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

3.4.2 Operational Feasibility

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

3.4.3 Technical Feasibility

The technical issue usually raised during the feasibility stage of the investigation includes the following

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipments have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?

- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

4. SYSTEM REQUIREMENTS SPECIFICATION

4.1 Introduction:

The phase in the system development life cycle that has crucial importance is the phase of the system study and the problem formulation. It is said that unless we understand the problem thoroughly, he will create problems rather than solving it. This phase includes the study of the existing system.

The first step in planning a software project is to prepare a concise statement of the problems to be solved and the constraints that exist for it solution/The problem statement should include a description of the present situation and goals to be achieved by the new system. The purpose is to get an idea about the operations and the interactions of the system, its drawbacks and to identify further requirements.

The analyst has to use various techniques to study the existing system. This includes interview of the personal related to the system, study of the literature available about the existing system, observation of the day to day activities ,he has to see whether it is feasible to replace the existing system.

***Interview*:**

One of the common most of obtaining data is interviewing the concerned persons. An interview is held to gather data on some areas. The coverage of the interview depends on the interviewed and the functions performed by that function.

Observation:

Observing the functioning of the system gives an understanding of the system. It also validates data gathered by various other means.

Research:

The study of available material is an important part of the data collections. Sources include company records of the system including operating records and data.

4.2 Purpose:

All these proposals are designed for non-mobile cloud environment. They consume large amount of storage and computation resources, which are not available for mobile devices. According to the experimental results in the basic ABE operations take much longer time on

mobile devices than laptop or desktop computers. It is at least 27 times longer to execute on a smart phone than a personal computer (PC). This means that an encryption operation which takes one minute on a PC will take about half an hour to finish on a mobile device. Furthermore, current solutions don't solve the user privilege change problem very well. Such an operation could result in very high revocation cost. This is not applicable for mobile devices as well. Clearly, there is no proper solution which can effectively solve the secure data sharing problem in mobile cloud. As the mobile cloud becomes more and more popular, providing an efficient secure data sharing mechanism in mobile cloud is in urgent need.

4.3 Functional Requirements

(1) Data Owner (DO): DO uploads data to the mobile cloud and share it with friends. DO determines the access control policies.

(2) Data User (DU): DU retrieves data from the mobile cloud.

(3) Trust Authority (TA): TA is responsible for generating and distributing attribute keys.

(4) Encryption Service Provider (ESP): ESP provides data encryption operations for DO.

(5) Decryption Service Provider (DSP): DSP provides data decryption operations for DU.

(6) Cloud Service Provider (CSP): CSP stores the data for DO. It faithfully executes the operations requested by DO, while it may peek over data that DO has stored in the cloud.

4.4 Non-Functional Requirements

The major non-functional Requirements of the system are as follows

Usability

The system is designed with completely automated process hence there is no or less user intervention.

Reliability

The system is more reliable because of the qualities that are inherited from the chosen platform java. The code built by using java is more reliable.

Performance

This system is developing in the high level languages and using the advanced front-end and back-end technologies it will give response to the end user on client system with in very less time.

Supportability

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is having JVM, built into the system.

Implementation

The system is implemented in web environment using struts framework. The apache tomcat is used as the web server and windows xp professional is used as the platform Interface the user interface is based on Struts provides HTML Tag

4.5 System Requirements:

Hardware Required	**:**	
System	:	Pentium IV 2.4 GHz
Hard Disk	:	40 GB
Floppy Drive	:	1.44 MB
Monitor	:	15 VGA color
Mouse	:	Logitech
Keyboard	:	110 Keys enhanced
RAM	:	256MB
Software Required	**:**	
Operating System	:	Windows 7,8,10
Language	:	Java (jdk8.0)
IDE	:	Net Beans 8.0
Back-End	:	MY SQL

5. SYSTEM DESIGN

5.1 System Specification:

The objective of this sub-project is to develop tools and methods to support the earlier phases of systems development for implementation independent specification and verification, and for subsequent synthesis of specifications into efficient implementations.

The sub-project is divided into four sub-tasks:

- adopt/further develop a model for formal, high-level system specification and verification.
- Demonstrate the efficacy of the developed model by applying it to a suitable part of the consortium demonstrator, the network terminal for broadband access.
- Develop a systematic method to refine the specification into synthesizable code and a prototype tool which supports the refinement process and links it to synthesis and compilation tools.

Justification

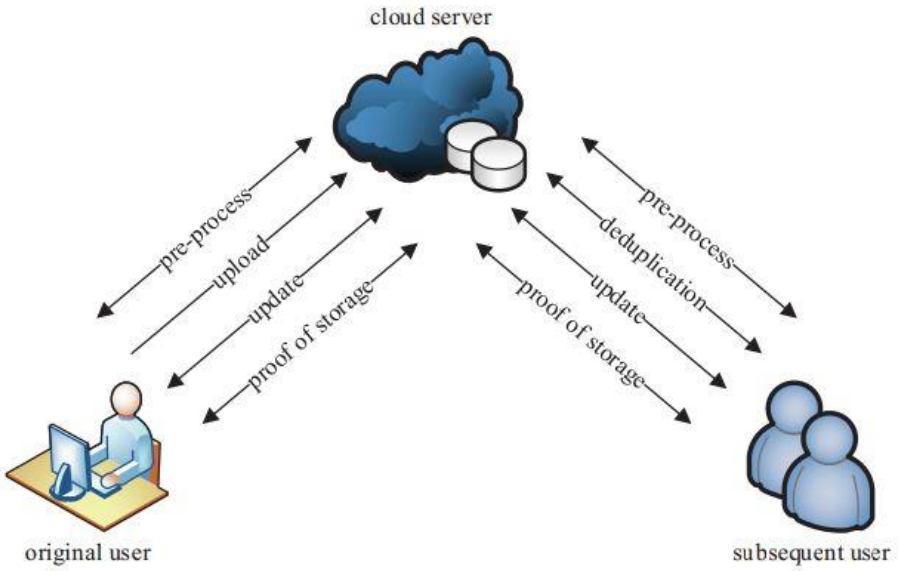
Today it is common to specify systems on higher levels using some natural language (e.g. English). For large systems, where large amounts of information must be handled, problems arise with ambiguities and inconsistencies with such specifications. Errors that are introduced are often detected late in the design cycle - in the simulation of the design after much design work has already been carried out - if detected at all.

By making the initial system specifications in a formal language at a high abstraction level, functionality can be verified/simulated earlier in the development process. Ambiguities and in consistencies can be avoided, errors can be discovered earlier, and the design iteration cycles can be shortened, thereby reducing development times. It is of critical importance that the specification language provides modeling concepts at a high abstraction level to allow the representation of system functions at a conceptual level without introducing unnecessary details.

Further, most of the languages that are used for implementation of HW / SWdesigns (e.g. VHDL, C++) do not lend themselves well to formal verification. This is because they lack a formally defined semantics or because the semantics is complex. A lack of formal semantics sometimes causes ambiguities in the interpretation of the designs.

Our goal is to develop functional system specification method for telecom systems, to demonstrate its efficacy on an industrially relevant example and to develop a tool to support the mapping of such specifications to synthesizable VHDL/C++. The specification language in which the system level functions will be developed will have a formal semantics in order to support formal verification of specifications.

Architecture:



5.2 UML Diagrams:

5.2.1 Use case Diagram:

Overview

To model a system the most important aspect is to capture the dynamic behavior. To clarify a bit in details, dynamic behavior means the behavior of the system when it is running operating. So only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction. These internal and external agents are known as actors. So use case diagrams are consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem

of an application. A single use case diagram captures a particular functionality of a system. So to model the entire system numbers of use case diagrams are used.

Purpose

The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified. Now when the initial task is complete use case diagrams are modeled to present the outside view.

So in brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Show the interacting among the requirements are actors.

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So it can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system. The actors can be human user, some internal applications or may be some external applications. So in a brief when we are planning to draw a use case diagram, it should have the following items identified.

- Functionalities to be represented as an use case
- Actors
- Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. So after identifying the above items the following guidelines to draw an efficient use case diagram.

- The name of a use case is very important. So the name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.

- Use note when ever required to clarify some important points.

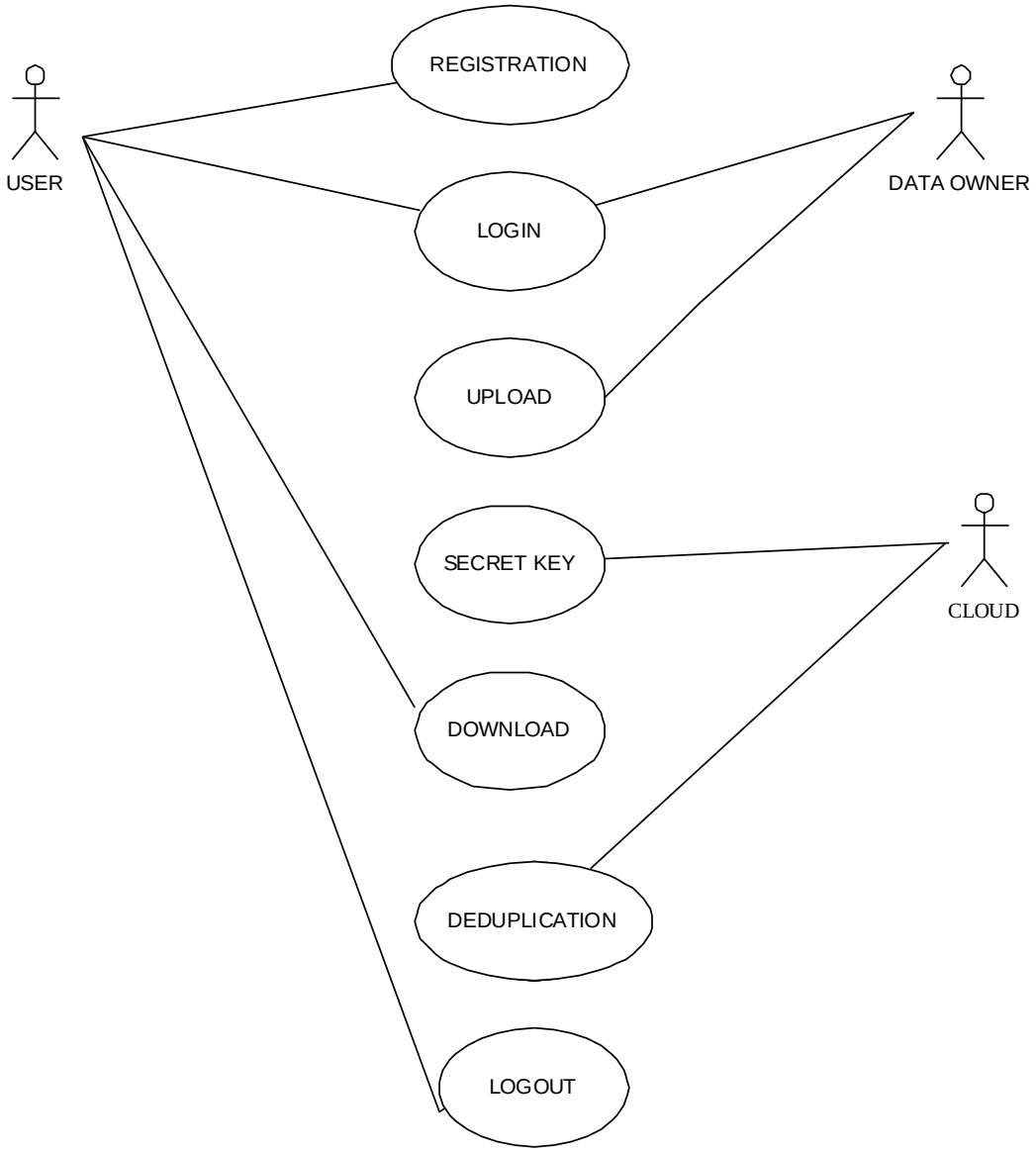


Fig 5.2.1 Use case Diagram

5.2.2 Class Diagram:

Overview

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.

The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the

modeling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages. The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram.

Purpose

The purpose of the class diagram is to model the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction. The UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application but class diagram is a bit different. So it is the most popular UML diagram in the coder community.

So the purpose of the class diagram can be summarized as:

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

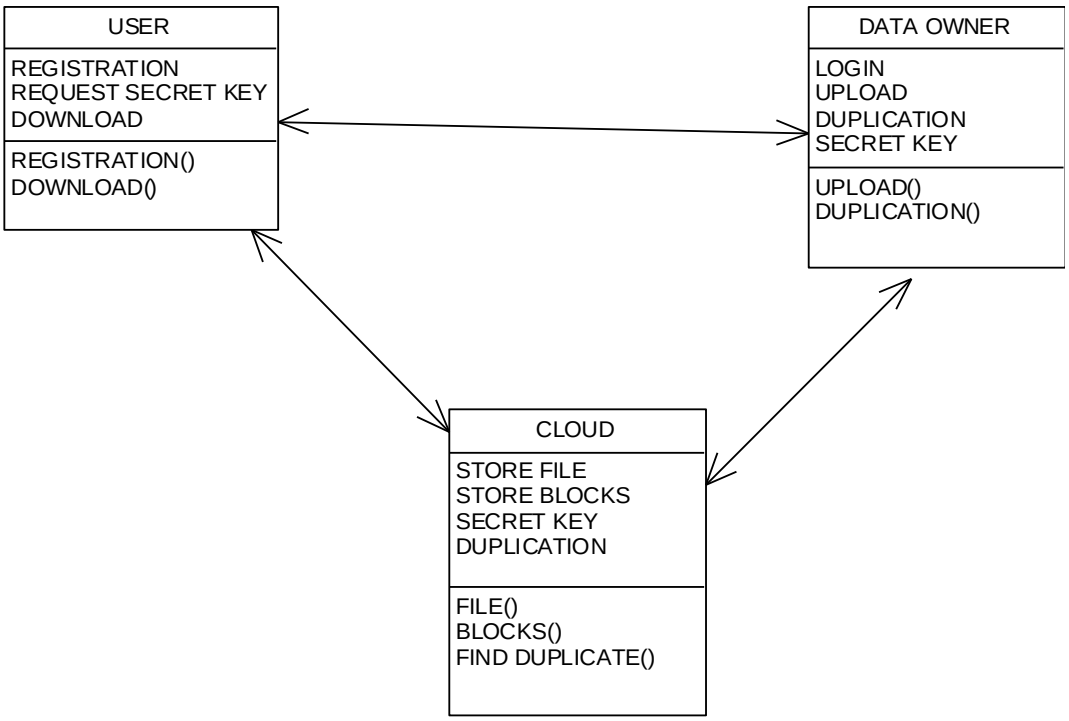


Fig 5.2.2 Class Diagram

5.2.3 Sequence Diagram:

Overview

From the name Interaction it is clear that the diagram is used to describe some type of interactions among the different elements in the model. So this interaction is a part of dynamic behavior of the system. This interactive behavior is represented in UML by two diagrams known as Sequence diagram and Collaboration diagram. The basic purposes of both the diagrams are similar. Sequence diagram emphasizes on time sequence of messages and collaboration diagram emphasizes on the structural organization of the objects that send and receive messages.

Purpose

The purposes of interaction diagrams are to visualize the interactive behavior of the system. Now visualizing interaction is a difficult task. So the solution is to use different types of models to capture the different aspects of the interaction. That is why sequence and collaboration diagrams are used to capture dynamic nature but from a different angle. So the purposes of interaction diagram can be describes as:

- To capture dynamic behavior of a system.
- To describe the message flow in the system.
- To describe structural organization of the objects.
- To describe interaction among objects.

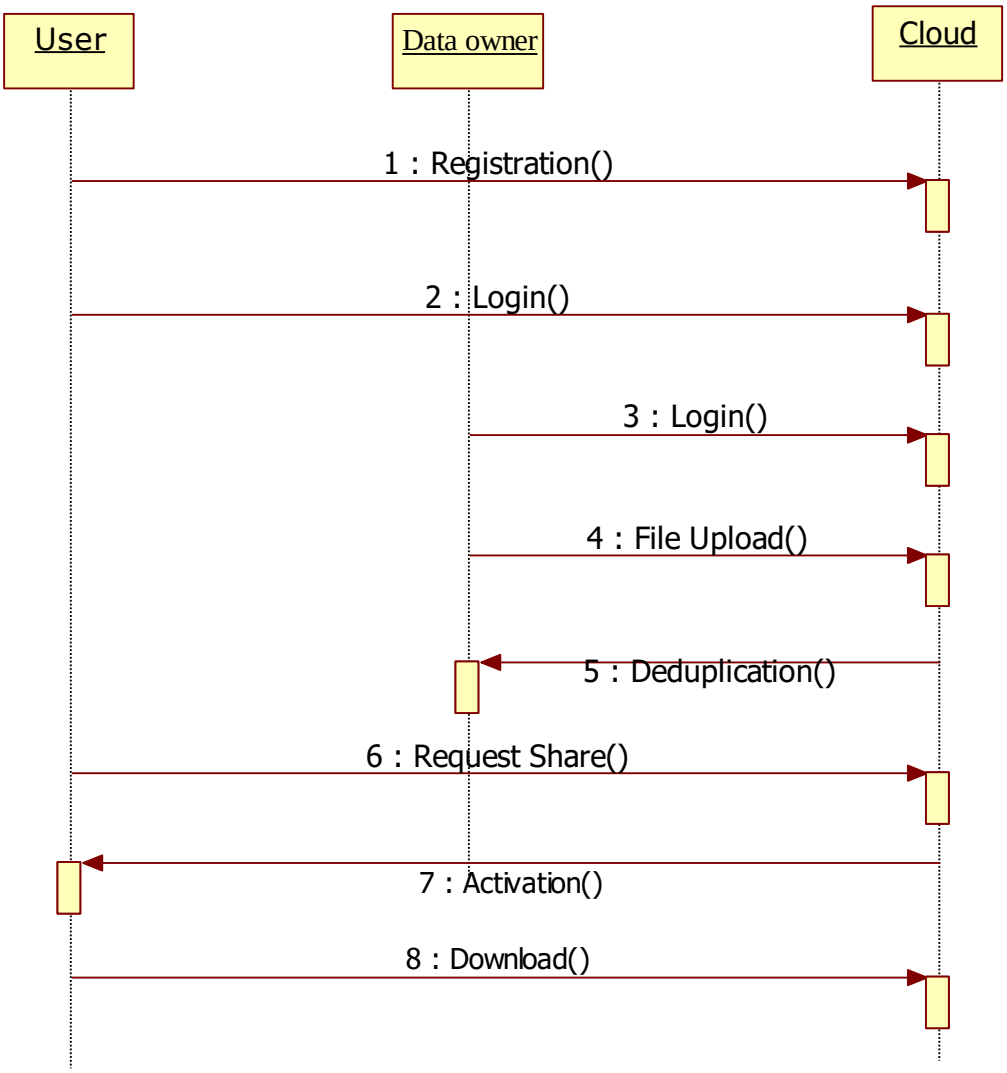


Fig 5.2.3 Sequence Diagram

5.2.4 Collaboration Diagram:

Overview

Class diagrams indicates what classes are part of the system, what they offer, how they relate, But they don't tell us how they communicate. Collaboration diagrams show (used to model) How objects interact and their roles. They are very similar to sequence diagrams Sequence Diagrams are arranged according to Time. Collaboration Diagrams represent the structural organization of object.

Collaboration diagrams are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaborations are used by designers to define and clarify the roles of the

objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces.

Unlike a sequence diagram, a collaboration diagram shows the relationships among the objects. Sequence diagrams and collaboration diagrams express similar information, but show it in different ways. Collaboration diagrams show the relationships among objects and are better for understanding all the effects on a given object and for procedural design.

Because of the format of the collaboration diagram, they tend to better suited for analysis activities. Specifically, they tend to be better suited to depicting simpler interactions of smaller numbers of objects. As the number of objects and messages grows, the diagram becomes increasingly hard to read. In addition, it is difficult to show additional descriptive information such as timing, decision points, or other unstructured information that can be easily added to the notes in a sequence diagram.

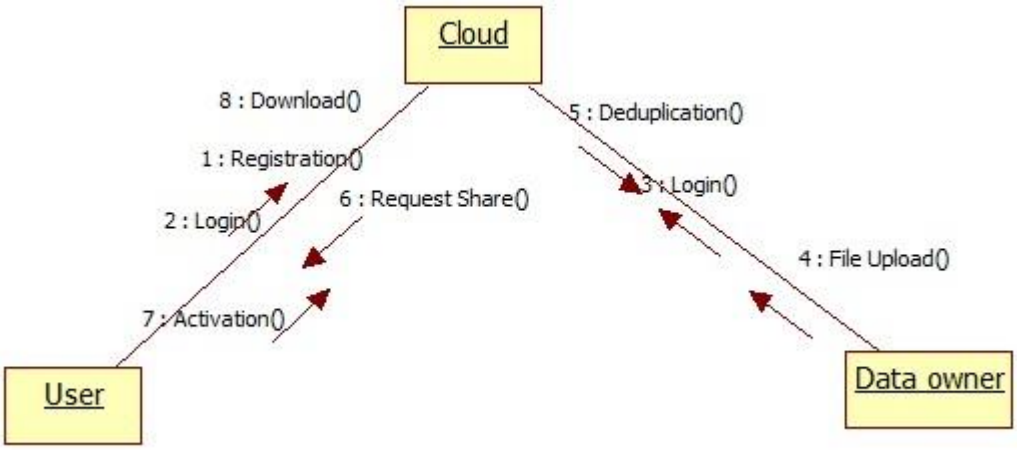


Fig 5.2.4 Collaboration Diagram

5.2.5 State Chart Diagram:

Overview

A State chart diagram describes a state machine. Now to clarify it state machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

Purpose

State chart diagram is one of the five UML diagrams used to model dynamic nature of a system. They define different states of an object during its lifetime. And

these states are changed by events. So the State chart diagrams are useful to model reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. So the most important purpose of State chart diagram is to model life time of an object from creation to termination. State chart diagrams are also used for forward and reverse engineering of a system. But the main purpose is to model reactive system.

Following are the main purposes of using State chart diagrams

- To model dynamic aspect of a system.
- To model life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model states of an object.

State chart diagram is used to describe the states of different objects in its life cycle. So the emphasis is given on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately. State chart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.

Before drawing a State chart diagram we must have clarified the following points:

- Identify important objects to be analyzed.
- Identify the states.

Identify the events.

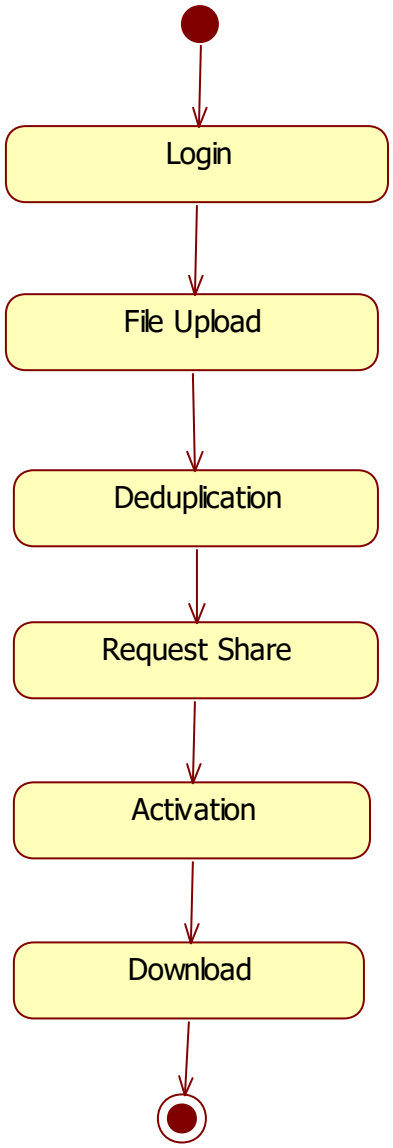


Fig 5.2.5 State Chart Diagram

5.2.6 Activity Diagram:

Overview

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

Purpose

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another. Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

So the purposes can be described as

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

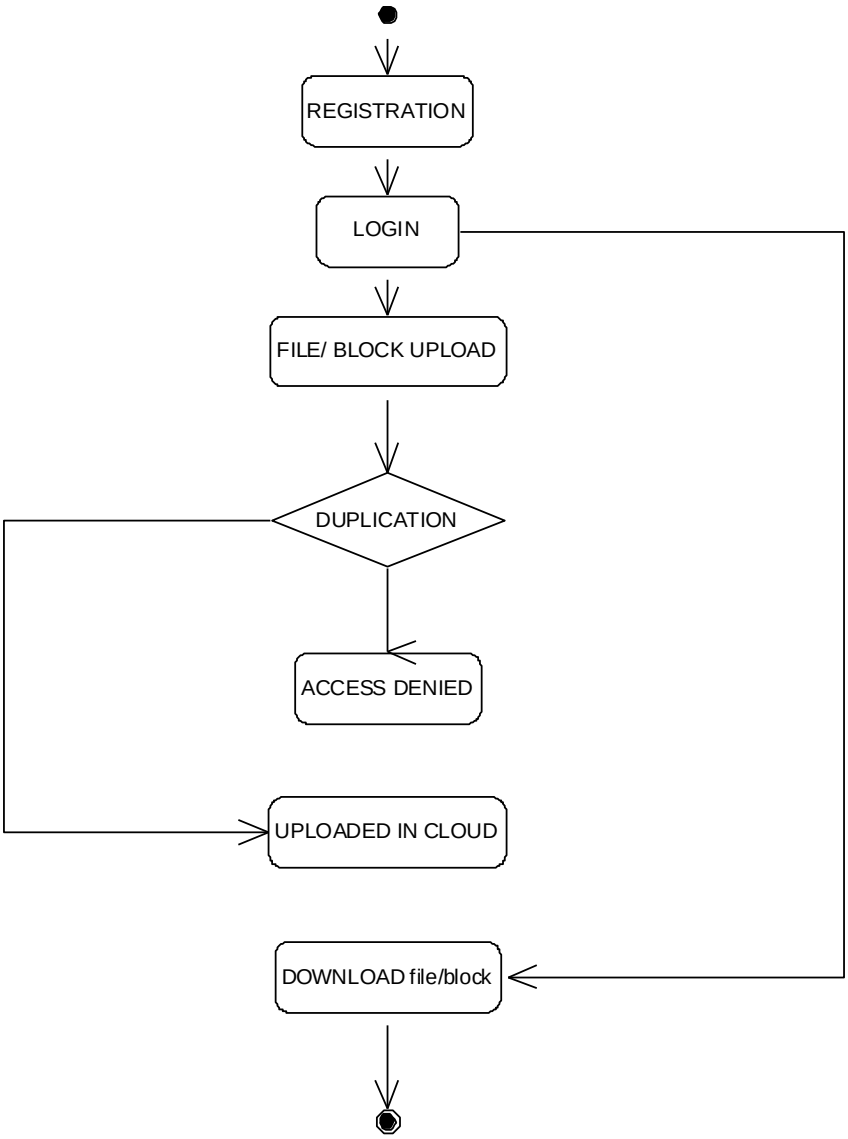


Fig 5.2.6 Activity Diagram

5.2.7 Component Diagram:

Overview

Component diagrams are different in terms of nature and behavior. Component diagrams are used to model physical aspects of a system. Physical aspects are the elements like executables, libraries, files, documents etc which resides in a node. So component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

Purpose

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the

functionality of the system but it describes the components used to make those functionalities. So from that point component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files etc. Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment. A single component diagram cannot represent the entire system but a collection of diagrams are used to represent the whole.

So the purpose of the component diagram can be summarized as:

- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.

Describe the organization and relationships of the components.

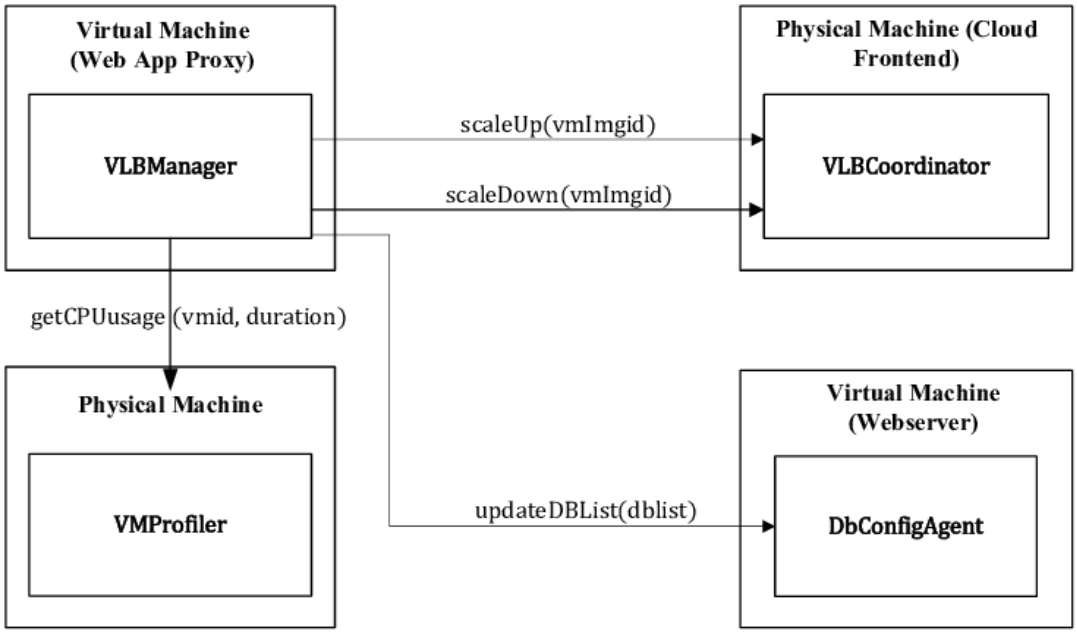


Fig 5.2.7 Component Diagram

5.3 Modules Description:

- System Framework
- Data Owner
- Data User
- Trusted Authority
- Cloud Service Provider

MODULES DESCRIPTION:

System Framework:

The development of cloud computing and the popularity of smart mobile devices, people are gradually getting accustomed to a new era of data sharing model in which the data is stored on the cloud and the mobile devices are used to store/retrieve the data from the cloud.In these applications, people (dataowners) can upload their documents andother files to the cloud and share these data with otherpeople (data users) they like to share. CSPs also providedata management functionality for data owners. Sincepersonal data files are sensitive, data owners are allowedto choose whether to make their data files public or canonly be shared with specific data users. Clearly, dataprivacy of the personal sensitive data is a big concern formany data owners.We propose LDSS, a framework of lightweight datasharing scheme in mobile cloud. It has thefollowing six components. (1)Data Owner (DO) (2) Data User (DU) (3) Trust Authority (TA) (4) Encryption Service Provider (ESP) (5) Decryption Service Provider (DSP) (6) Cloud Service Provider (CSP).

Data Owner (DO):

When the data owner (DO) registers on TA, TA runs the algorithm Setup() to generate a public key PK and a master key MK. PK is sent to DO while MK is kept on TA itself. DO defines its own attribute set and assignsattributes to its contacts. All these information will be sent to TA and the cloud.TA and the cloud receive the information and storeit.DO uploads data to the mobilecloud and share it with friends. DO determines the accesscontrol policies.DO sends data to the cloud. Sincethe cloud is not credible, data has to be encrypted before it is uploaded. The DO defines access

control policy in the form of access control tree on data files to assign which attributes a DU shouldobtain if he wants to access a certain data file.

Data User (DU):

DU logins onto the system and sends, an authorization request to TA. The authorization request includes attribute keys (SK) which DU already has.TA accepts the authorization request and checks the request and a generate attribute keys (SK)for DU.DU sends a request for data to the cloud.Cloud receives the request and checks if the DU meets the access requirement. DU receives the ciphertext, which includesciphertext of data files and ciphertext of the symmetric key. DU decrypt the ciphertext of the symmetric key with the assistance of DSP. DU uses the symmetric key to decrypt the ciphertext of data files.

Trusted Authority:

To make LDSS feasible in practice, a trustedauthority (TA) is introduced. It is responsible ofgenerating public and private keys, and distributingattribute keys to users. With this mechanism, users can share and access data without being aware of the encryption and decryption operations. We assume TA is entirely credible, and a trusted channel exists between the TA and every user. The fact that a trusted channel exists doesn't mean that the data can be shared through the trusted channel, for the data can be in a large amount. TA is only used to transfer keys (in a small amount) securely between users. In addition, it's requested that TA is online all the time because data users may access data at any time and need TA to update attribute keys.

Cloud Service Provider:

CSP stores the data for DO. It faithfully executes the operations requested by DO, while it may peek over data that DO has stored in the cloud.DU sends a request for data to the cloud. Cloud receives the request and checks if the DU meets the access requirement. If DU can't meet the requirement, it refuses therequest; otherwise it sends the ciphertext to DU. CSP manages the Uploaded Files.

6. IMPLEMENTATION

6.1 Technology Description:

About Java Plate form

The Java platform consists of the Java application programming interfaces (APIs) and the Java virtual machine (JVM).

Fig 6.1 Software testing life cycle

The following Java technology lets developers, designers, and business partners develop and deliver a consistent user experience, with one environment for applications on mobile and embedded devices. Java meshes the power of a rich stack with the ability to deliver customized experiences across such devices.

Java APIs are libraries of compiled code that you can use in your programs. They let you add ready-made and customizable functionality to save you programming time .Java programs are run (or interpreted) by another program called the Java Virtual Machine. Rather than running directly on the native operating system, the program is interpreted by the Java VM for the native operating system. This means that any computer system with the Java VM installed can run Java programs regardless of the computer system on which the applications were originally developed.

In the Java programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the javac compiler. A .class file does not contain code that is native to your processor; it instead contains byte codes the machine language of the Java Virtual Machine (Java VM). The java launcher tool then runs your application with an instance of the Java Virtual Machine.

Because the Java VM is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris TM Operating System (Solaris OS), Linux, or Mac OS.

Java technology is both a programming language and a platform.

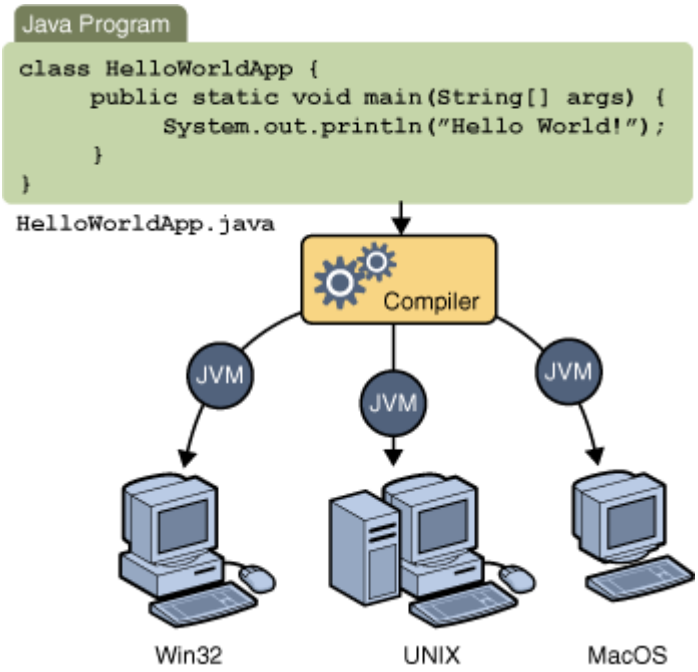


Fig 6.2 Java Program For Execution

Through the Java VM, the same application is capable of running on multiple platforms.

What is a Java Virtual Machine

To understand the Java virtual machine you must first be aware that you may be talking about any of three different things when you say "Java virtual machine." You may be speaking of:

- the abstract specification
- a concrete implementation
- A runtime instance.

The abstract specification is a concept, described in detail in the book: The Java Virtual Machine Specification, by Tim Lindholm and Frank Yellin. Concrete implementations, which exist on many platforms and come from many vendors, are either all software or a combination of hardware and software. A runtime instance hosts a single running Java application.

Each Java application runs inside a runtime instance of some concrete implementation of the abstract specification of the Java virtual machine. In this book, the term "Java virtual machine" is used in all three of these senses. Where the intended sense is not clear from the context, one of the terms "specification," "implementation," or "instance" is added to the term "Java virtual machine".

The Architecture of the Java Virtual Machine

In the Java virtual machine specification, the behavior of a virtual machine instance is described in terms of subsystems, memory areas, data types, and instructions. These components describe an abstract inner architecture for the abstract Java virtual machine. The purpose of these components is not so much to dictate an inner architecture for implementations. It is more to provide a way to strictly define the external behavior of implementations. The specification defines the required behavior of any Java virtual machine implementation in terms of these abstract components and their interactions.

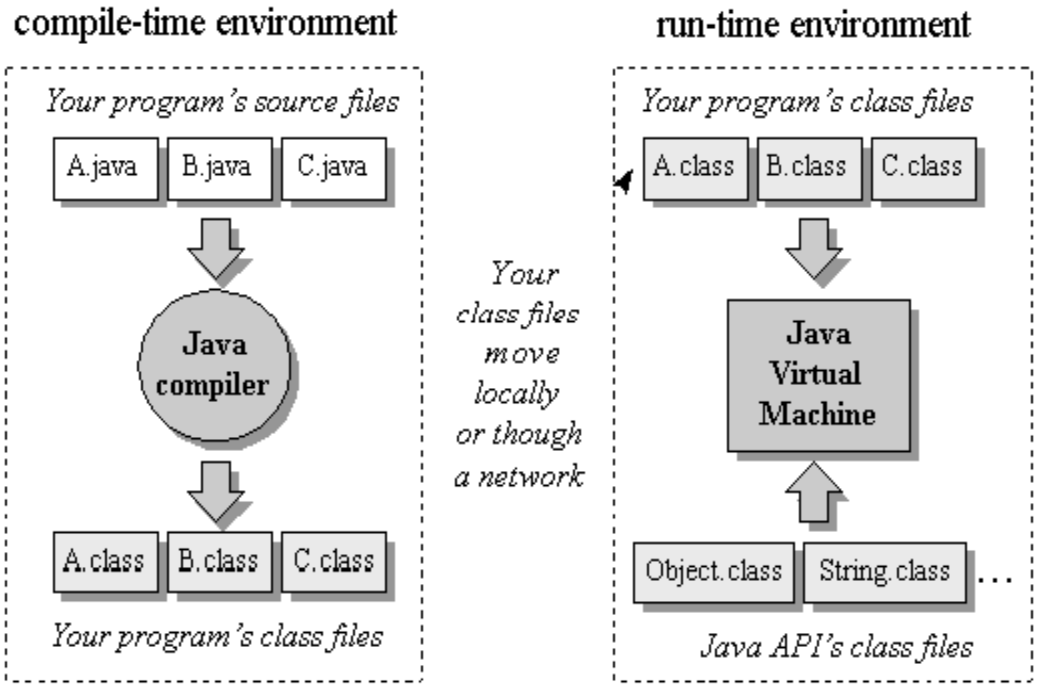


Fig 6.3 Java virtual machine

shows a block diagram of the Java virtual machine that includes the major subsystems and memory areas described in the specification. As mentioned in previous chapters, each Java virtual machine has a class loader subsystem: a mechanism for loading types (classes and interfaces) given fully qualified names.

Each Java virtual machine also has an execution engine: a mechanism responsible for executing the instructions contained in the methods of loaded classes.

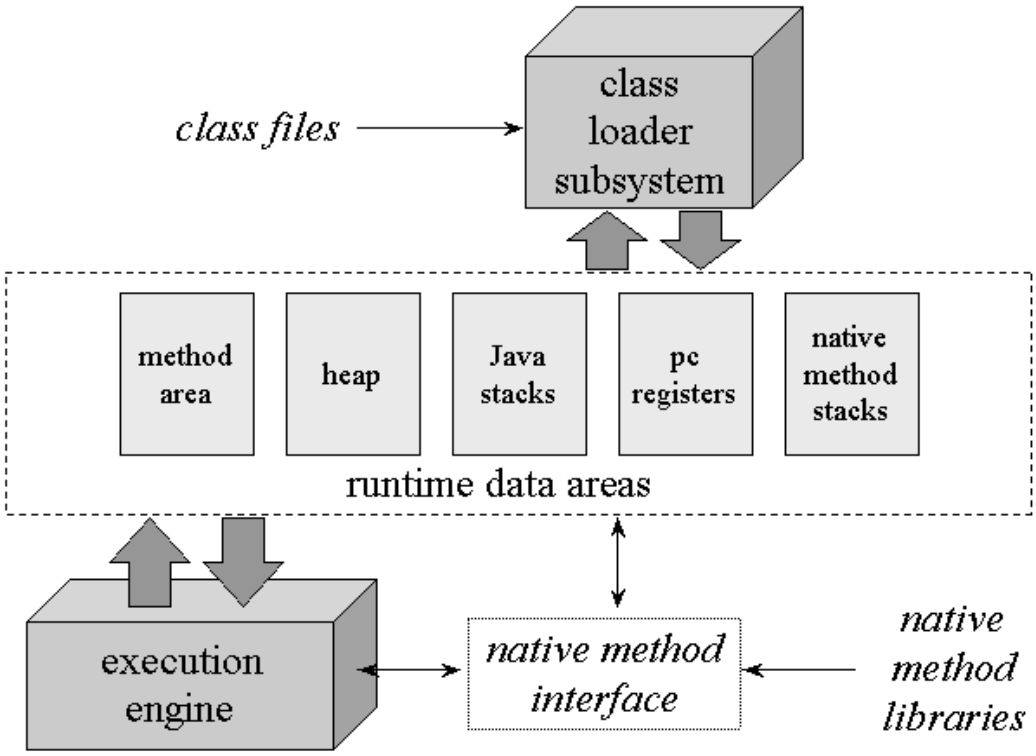


Fig 6.4 The internal architecture of the Java virtual machine

When a Java virtual machine runs a program, it needs memory to store many things, including bytecodes and other information it extracts from loaded class files, objects the program instantiates, parameters to methods, return values, local variables, and intermediate results of computations. The Java virtual machine organizes the memory it needs to execute a program into several runtime data areas.

Although the same runtime data areas exist in some form in every Java virtual machine implementation, their specification is quite abstract. Many decisions about the structural details of the runtime data areas are left to the designers of individual implementations.

Different implementations of the virtual machine can have very different memory constraints. Some implementations may have a lot of memory in which to work, others may have very little. Some implementations may be able to take advantage of virtual memory, others may not. The abstract nature of the specification

of the runtime data areas helps make it easier to implement the Java virtual machine on a wide variety of computers and devices.

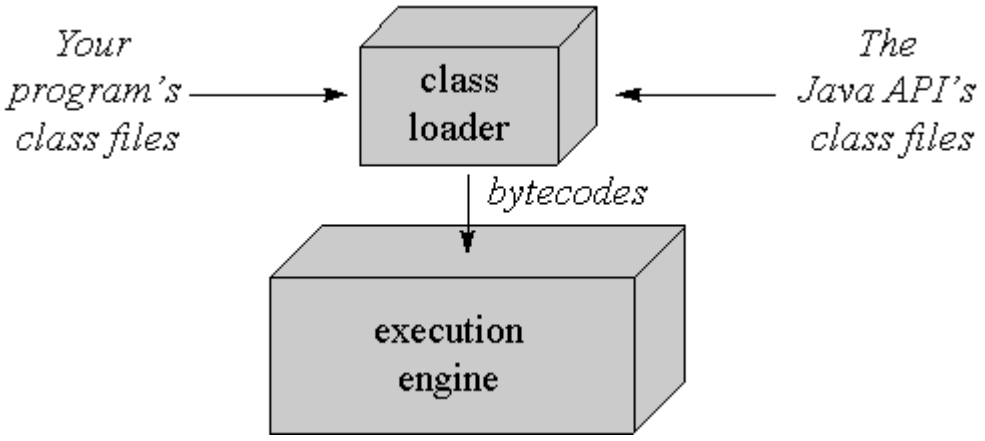


Fig 6.5 Runtime program

Some runtime data areas are shared among all of an application's threads and others are unique to individual threads. Each instance of the Java virtual machine has one method area and one heap. These areas are shared by all threads running inside the virtual machine. When the virtual machine loads a class file, it parses information about a type from the binary data contained in the class file. It places this type information into the method area. As the program runs, the virtual machine places all objects the program instantiates onto the heap for a graphical depiction of these memory areas.

As each new thread comes into existence, it gets its own pc register (program counter) and Java stack. If the thread is executing a Java method (not a native method), the value of the pc register indicates the next instruction to execute. A thread's Java stack stores the state of Java (not native) method invocations for the thread. The state of a Java method invocation includes its local variables, the parameters with which it was invoked, its return value (if any), and intermediate calculations. The state of native method invocations is stored in an implementation-dependent way in native method stacks, as well as possibly in registers or other implementation-dependent memory areas

The Java stack is composed of stack frames (or frames). A stack frame contains the state of one Java method invocation. When a thread invokes a method, the Java virtual machine pushes a new frame onto that thread's Java stack. When the method completes, the virtual machine pops and discards the frame for that method.
The Java virtual machine has no registers to hold intermediate data values. The instruction set uses the Java stack for storage of intermediate data values. This

approach was taken by Java's designers to keep the Java virtual machine's instruction set compact and to facilitate implementation on architectures with few or irregular general purpose registers. In addition,

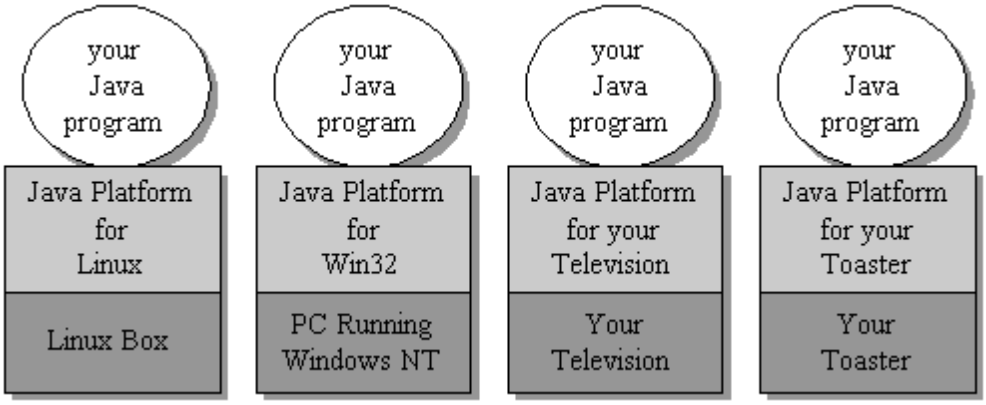


Fig6.6 Stack-based architecture

the stack-based architecture of the Java virtual machine's instruction set facilitates the code optimization work done by just-in-time and dynamic compilers that operate at run-time in some virtual machine implementations.

See for a graphical depiction of the memory areas the Java virtual machine creates for each thread. These areas are private to the owning thread. No thread can access the pc register or Java stack of another thread.

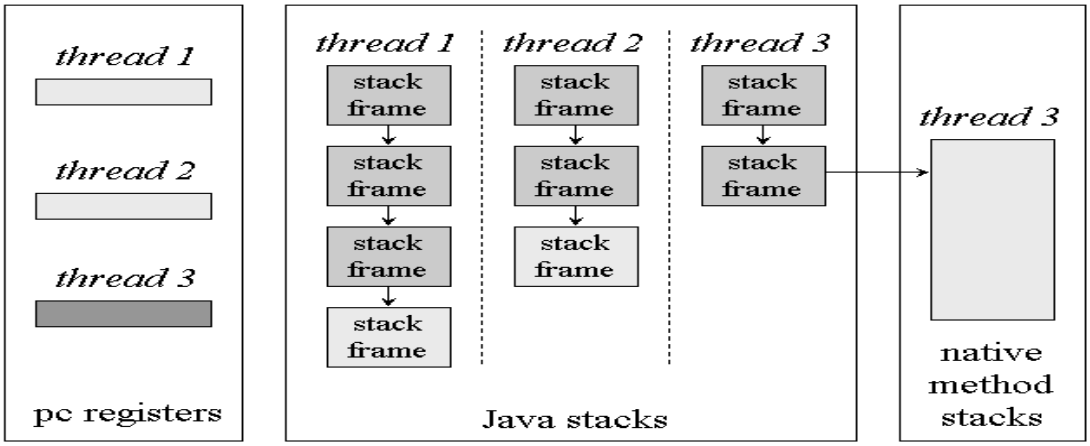


Fig6.7 Runtime data areas exclusive to each thread

Figure shows a snapshot of a virtual machine instance in which three threads are executing. At the instant of the snapshot, threads one and two are executing Java methods. Thread three is executing a native method.

In Figure, as in all graphical depictions of the Java stack in this book, the stacks are shown growing downwards. The "top" of each stack is shown at the bottom of the figure. Stack frames for currently executing methods are shown in a lighter shade. For threads that are currently executing a Java method, the pc register indicates the next instruction to execute. In such pc registers (the ones for threads one and two) are shown in a lighter shade. Because thread three is currently executing a native method, the contents of its pc register--the one shown in dark gray--is undefined.

JVM is an Emulation

The difficult part of creating Java byte code is that the source code is compiled for a machine that does not exist. This machine is called the Java Virtual Machine, and it exists only in the memory of our computer. Fooling the Java compiler into creating byte code for a nonexistent machine is only one-half of the ingenious process that makes the Java architecture neutral. The Java interpreter must also make our computer and the byte code file believe they are running on a real machine. It does this by acting as the intermediary between the Virtual Machine and our real machine.

The Java Virtual Machine is responsible for interpreting Java byte code and translating this into actions or Operating System calls. For example, a request to establish a socket connection to a remote machine will involve an Operating System call Different Operating Systems handle sockets in different ways - but the programmer doesn't need to worry about such details. It is the responsibility of the JVM to handle these translations so that the Operating System and the CPU architecture on which the Java software is running is completely irrelevant to the developer.

The Basic Parts of the Java Virtual Machine

Creating a Virtual Machine within our computer's memory requires building every major function of a real computer down to the very environment within which programs operate. These functions can be broken down into seven basic parts:

- A set of registers
- A stack
- An execution environment
- A garbage-collected heap
- A constant pool
- A method storage area
- An instruction set

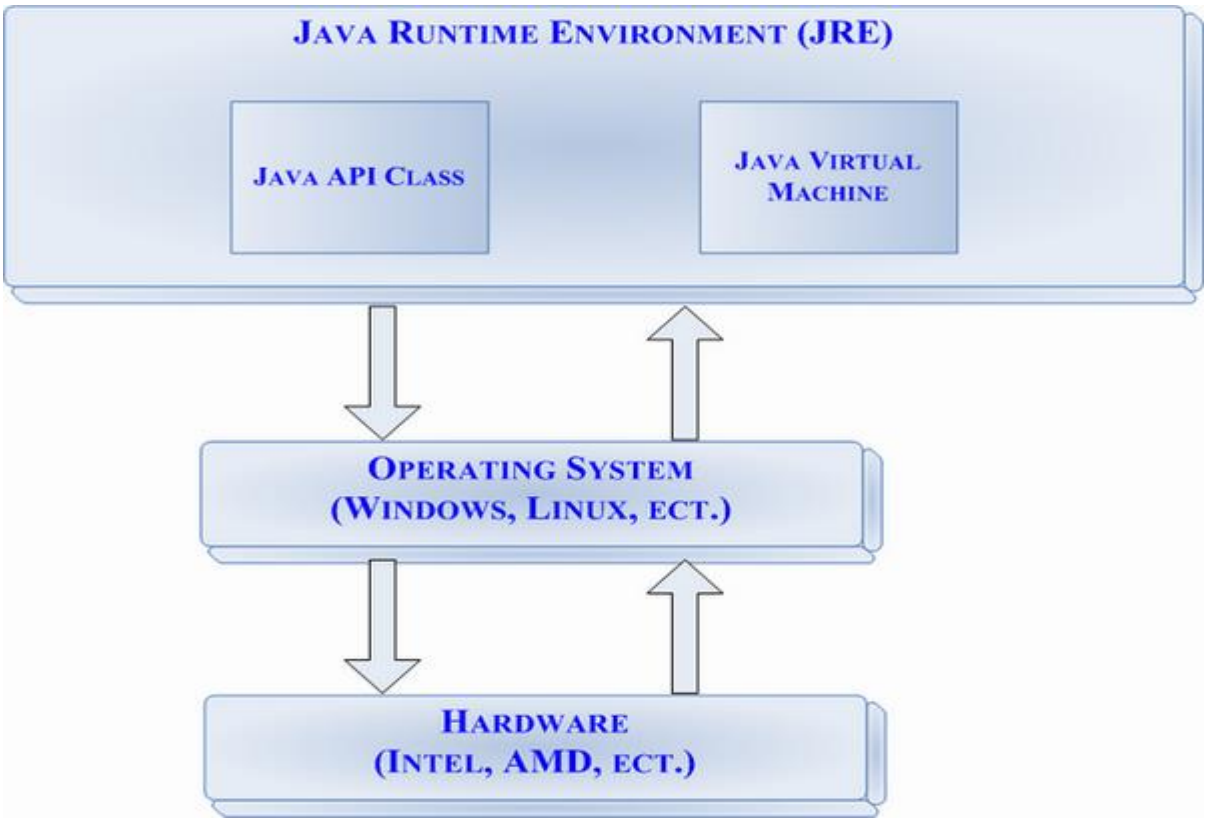


Fig 6.8 JVM handles translations

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 written in C++.
- **Write better code**: The Java programming language encourages good coding practices, and automatic garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans™ component architecture, and its wide-

ranging, easily extendible API let you reuse existing, tested code and introduce fewer bugs.

- **Develop programs more quickly**: The Java programming language is simpler than C++, and as such, your development time could be up to twice as fast when writing in it. Your programs will also require fewer lines of code.
- **Avoid platform dependencies**: You can keep your program portable by avoiding the use of libraries written in other languages.
- **Write once, run anywhere**: Because applications written in the Java programming language are compiled into machine-independent bytecodes, they run consistently on any Java platform.
- **Distribute software more easily**: With Java Web Start software, users will be able to launch your applications with a single click of the mouse. An automatic version check at startup ensures that users are always up to date with the latest version of your software. If an update is available, the Java Web Start software will automatically update their installation.

An Overview of the AWT

AWT stands for Abstract Window Tool Kit. The Abstract Window Toolkit supports GUI Java programming. It is a portable GUI library for stand-alone applications and/or applets. The Abstract Window Toolkit provides the connection between your application and the native GUI. The AWT provides a high level of abstraction for your Java program since it hides you from the underlying details of the GUI your program will be running on.

AWT features include:

- A rich set of user interface components.
- A robust event-handling model.
- Graphics and imaging tools, including shape, color, and font classes.
- Layout managers, for flexible window layouts that don't depend on a particular window size or screen resolution.
- Data transfer classes, for cut-and-paste through the native platform clipboard.

The AWT components depend on native code counterparts (called peers) to handle their functionality. Thus, these components are often called "heavyweight" components.

An Overview of Swing

Swing implements a set of GUI components that build on AWT technology and provide a pluggable look and feel. Swing is implemented entirely in the Java programming language, and is based on the JDK 1.1 Lightweight UI Framework. Swing features include:

- All the features of AWT.
- 100% Pure Java certified versions of the existing AWT component set (Button, Scrollbar, Label, etc.).
- A rich set of higher-level components (such as tree view, list box, and tabbed panes).
- Pure Java design, no reliance on peers.
- Pluggable Look and Feel.

Swing components do not depend on peers to handle their functionality. Thus, these components are often called "lightweight" components

AWT vs. Swing

There are, of course, both pros and cons to using either set of components from the JFC in your Java applications. Here is a summary:

Pros

- Speed: use of native peers speeds component performance.
- Applet Portability: most Web browsers support AWT classes so AWT applets can run without the Java plugin.
- Look and Feel: AWT components more closely reflect the look and feel of the OS they run on

Cons

- Portability: use of native peers creates platform specific limitations. Some components may not function at all on some platforms.
- Third Party Development: the majority of component makers, including Borland and Sun, base new component development on Swing components. There is a much smaller set of AWT components available, thus placing the burden on the programmer to create his or her own AWT-based components.
- Features: AWT components do not support features like icons and tool-tips.

Swing

Pros

- Portability: Pure Java design provides for fewer platform specific limitations.

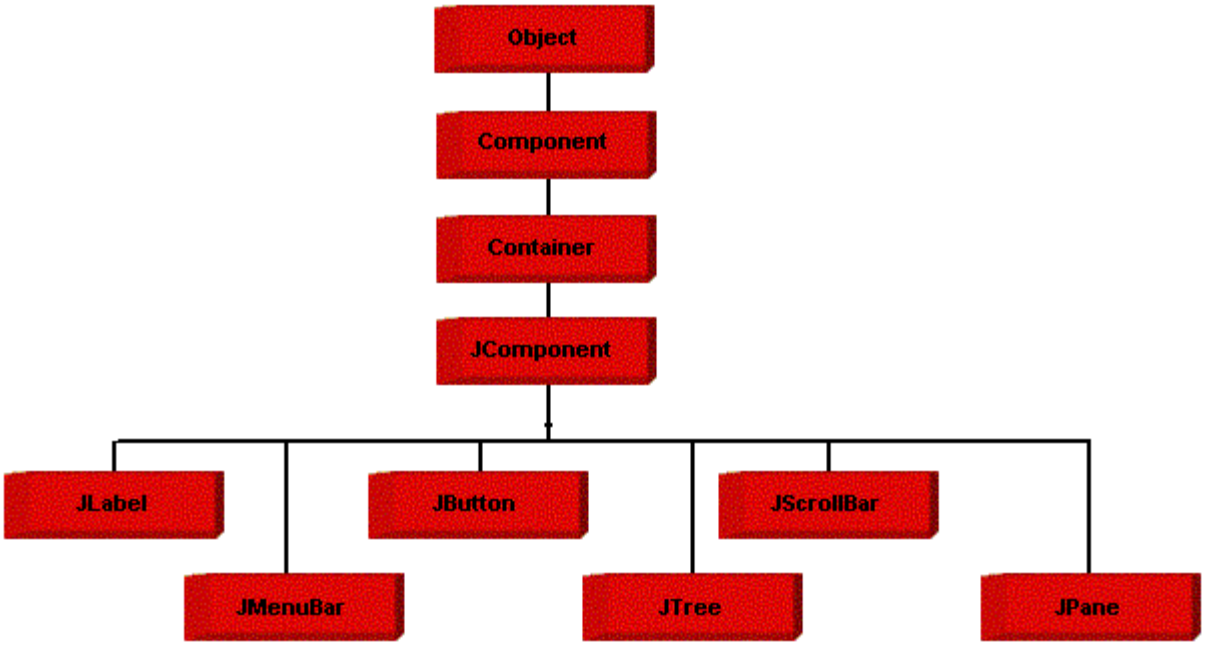
- Behavior: Pure Java design allows for a greater range of behavior for Swing components since they are not limited by the native peers that AWT uses.
- Features: Swing supports a wider range of features like icons and pop-up tool-tips for components.
- Vendor Support: Swing development is more active. Sun puts much more energy into making Swing robust.
- Look and Feel: The pluggable look and feel lets you design a single set of GUI components that can automatically have the look and feel of any OS platform (Microsoft Windows, Solaris, Macintosh, etc.). It also makes it easier to make global changes to your Java programs that provide greater accessibility (like picking a hi-contrast color scheme or changing all the fonts in all dialogs, etc.).

Cons

- Applet Portability: Most Web browsers do not include the Swing classes, so the Java plugin must be used.
- Performance: Swing components are generally slower and buggier than AWT, due to both the fact that they are pure Java and to video issues on various platforms. Since Swing components handle their own painting (rather than using native API's like DirectX on Windows) you may run into graphical glitches
- Look and Feel: Even when Swing components are set to use the look and feel of the OS they are run on, they may not look like their native counterparts.

Java Swing class hierarchy

The class J Component, descended directly from Container, is the root class for most of Swing's user interface components. Swing contains components that you'll use to build a GUI. I am listing you some of the commonly used Swing components. To learn and understand these swing programs, AWT Programming knowledge is not required.



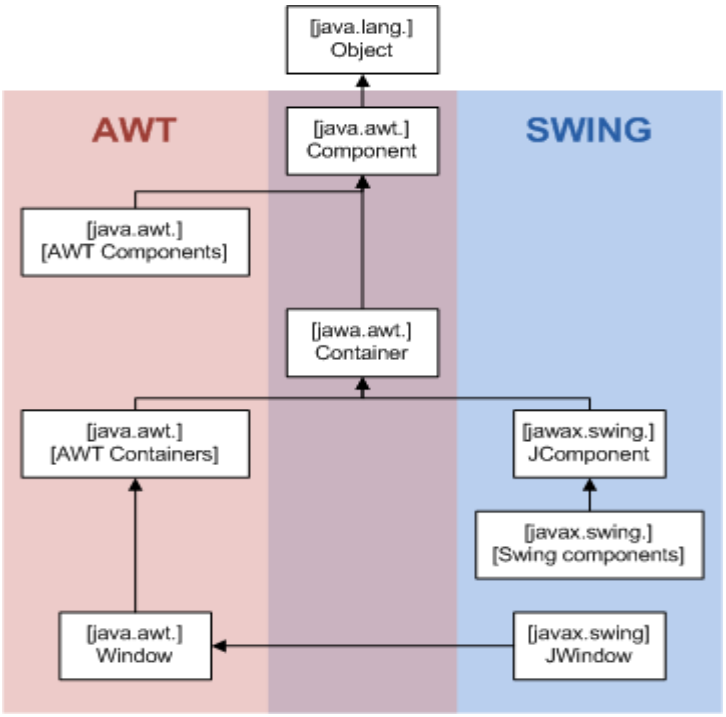
6.9 Swings GUI

Swing is built on top of AWT and is entirely written in Java, using AWT's lightweight component support. In particular, unlike AWT, t he architecture of Swing components makes it easy to customize both their appearance and behavior. Components from AWT and Swing can be mixed, allowing you to add Swing support to existing AWT-based programs. For example, swing components such as J Slider, J Button and J Checkbox could be used in the same program with standard AWT labels, text fields and scrollbars. You could subclass the existing Swing UI, model, or change listener classes without having to reinvent the entire implementation. Swing also has the ability to replace these objects on-the-fly.

Relationship to AWT and Swing

Since early versions of Java, a portion of the Abstract Window Toolkit (AWT) has provided platform-independent APIs for user interface components. In AWT, each component is rendered and controlled by a native peer component specific to the underlying windowing system By contrast Swing components are often described as lightweight because they do not require allocation of native resources in the operating system's windowing toolkit. The AWT components are referred to as heavyweight components.

Much of the Swing API is generally a complementary extension of the AWT rather than a direct replacement. In fact, every Swing lightweight interface ultimately exists within an AWT heavyweight component because all of the top-level components in Swing (J Applet, J Dialog, J Frame, and J Window) extend an AWT top-level container. Prior to Java 6 Update 10, the use of both lightweight and heavyweight components within the same window was generally discouraged due to Z-order incompatibilities. However, later versions of Java have fixed these issues, and both Swing and AWT components can now be used in one GUI without Z-order issues The core rendering functionality used by Swing to draw its lightweight components is provided by Java 2D, another part of JFC.



6.10 AWT and Swing Components

Below are some links to java swing tutorials that forms a helping hand to get started with java programming swing.

- J Panel is Swing's version of the AWT class Panel and uses the same default layout J Panel is descended directly from J Component.

- J Frame is Swing's version of Frame and is descended directly from that class. The components added to the frame are referred to as its contents; these are managed by the content Pane. To add a component to a JFrame, we must use its content Pane instead.

- J Internal Frame is confined to a visible area of a container it is placed in. It can be iconified maximized and layered.
- J Window is Swing's version of Window and is descended directly from that class. Like Window, it uses Border Layout by default.
- J Tabbed Pane contains a tab that can have a tool tip and a mnemonic, and it can display both text and an image.

6.2 Sample code:

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

/**
 *
 * @author java3
 */
import java.sql.Connection;
import java.sql.DriverManager;

/**
 *
 * @suresh java3
 */
public class DbConnection {
    public static Connection getConnection()
    {
        Connection con = null;
        try{
            Class.forName("com.mysql.jdbc.Driver");
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/lsdss",
"root", "root");
        }
        catch(Exception e)
        {
           e.printStackTrace();
        }
        return con;
    }
}
```

Algorithm code

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

/**
 *
 * @author java4
 */
import java.util.Scanner;
```

```
public class CaesarCipher
{
   public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz ";

   public static String encrypt(String filename, int shiftKey)
   {
      filename = filename.toLowerCase();
      String cipherText = "";
      for (int i = 0; i < filename.length(); i++)
      {
         int charPosition = ALPHABET.indexOf(filename.charAt(i));
         int keyVal = (shiftKey + charPosition) % 27;
         char replaceVal = ALPHABET.charAt(keyVal);
         cipherText += replaceVal;
      }
      return cipherText;
   }

   public static String decrypt(String filename, int shiftKey)
   {
      filename = filename.toLowerCase();
      String plainText = "";
      for (int i = 0; i < filename.length(); i++)
      {
         int charPosition = ALPHABET.indexOf(filename.charAt(i));
         int keyVal = (charPosition - shiftKey) % 27;
         if (keyVal < 0)
         {
            keyVal = ALPHABET.length() + keyVal;
         }
         char replaceVal = ALPHABET.charAt(keyVal);
         plainText += replaceVal;
      }
      return plainText;
   }

   public static void main(String[] args)
   {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the String for Encryption: ");
      String message = new String();
      message = sc.next();
      System.out.println(encrypt(message, 3));
      System.out.println(decrypt(encrypt(message, 3), 3));
      sc.close();
   }

   public Object encrypt(String name) {
      throw new UnsupportedOperationException("Not yet implemented");
   }
```

```
}
```

Split File

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

import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.RandomAccessFile;

/**
 *
 * @author java4
 */
public class SplitFile {

    public void split(String FilePath) throws IOException {
        RandomAccessFile raf = new RandomAccessFile(FilePath, "r");
        long numSplits = 3; //from user input, extract it from args
        long sourceSize = raf.length();
        long bytesPerSplit = sourceSize / numSplits;
        long remainingBytes = sourceSize % numSplits;
        int maxReadBufferSize = 1024; //8KB
        for (int destIx = 1; destIx <= numSplits; destIx++) {
            BufferedOutputStream bw = new BufferedOutputStream(new
FileOutputStream(FilePath + destIx));
            if (bytesPerSplit > maxReadBufferSize) {
                long numReads = bytesPerSplit / maxReadBufferSize;
                long numRemainingRead = bytesPerSplit % maxReadBufferSize;
                for (int i = 0; i < numReads; i++) {
                    readWrite(raf, bw, maxReadBufferSize);
                }
                if (numRemainingRead > 0) {
                    readWrite(raf, bw, numRemainingRead);
                }
            } else {
                readWrite(raf, bw, bytesPerSplit);
            }
            bw.close();
        }
//        if (remainingBytes > 0) {
//            BufferedOutputStream bw = new BufferedOutputStream(new
FileOutputStream("split." + (numSplits + 1)));
//            readWrite(raf, bw, remainingBytes);
//            bw.close()
```

7. SYSTEM TESTING

7.1 Introduction:

Software Testing is the process used to help identify the correctness, completeness, security and quality of developed computer software. Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate. In general, software engineers distinguish software faults from software failures. Our project" Visual cryptography For Cheating Prevention" is tested with the following testing methodologies.

7.2 Testing methods:

Developing Methodologies

The test process begins 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 for developing the test methodologies.

Acquire and study the test strategy

A team very familiar with the business risks associated with the software normally develops test strategy, the test team develops tactics. Thus the test team needs to acquire and study the test strategy. The test tactics are analyzed and studied for finding our various test factors, risks and effects. The risk involved in our project is implementing the encoding of the image. So, the proper knowledge about the testing strategies should be gained in order to avoid such high level risks.

Determine the type of development project

The type of the development refers to the platform or methodology for developing the project. As it is been a simulation project we go for the prototyping. The prototypes are simply predefined structure or model, which can be used for further modeling. By using the prototypes we can modify the existing module of the application for some other specific operations. Here the test tactics is to verify that all the tools are used properly and to test functionality.

Determine the type of software system

The type of software system relates to the type of processing which will be

encountered by the system. In this project, the software system we prefer to use is Java . We have chosen Java for its portability and its support to graphics & multimedia specifically for image processing.

Determine the scope of the software system

The scope of the project refers to the overall activities or operation to be included into the system being tested. The scope of the new system varies from that of the existing one. In the existing system, a large overhead occurs in contrast and pixel expansion. Also, the verification process is not efficient in the existing system. In this project, the pixel expansion is optimal because only two sub pixels are added each and every pixel. Also, each and every participants are verified or authentication.

Identify the tactical risks

The tactical risk is the subsets at a lower level of the strategic risks. The risks related to the application and its methodologies are identified. The risk involved in our project is implementing the encoding of the image.

Determine when the should occur testing

In the above processes we have identified the type of processing, scope and risks associated with our project. The testing can occur throughout all the phases of the project. During the analysis phase, the testing strategy and requirements are determined. In design phase, the complexities in design with respect to the requirements are determined and structural and functional test conditions are also tested. During implementation, the design consistency is determined. In test phase, the overall testing of the application is being done and previously the adequacy of the testing plan is also determined. In maintenance phase, the testing for modifying and reusing the system is done.

Build the system test plan

The test plan of the project should provide all the information on the application that is being tested. The test plan is simply a model that has to be followed during the progression of the testing. The test plan consists of the sequential set of procedures to test the application. Initially, the selection process of both secret and verification images are tested. Then the test is carried out for encoding of image, verification process and finally decoding process.

Build the unit test plan

In this case we are dividing the system into three different components or units each having specific functions. The three different components of the system are

browser window designing, browser events handling and adding speech to the browser. These units have their own test plan. The main purpose of the unit test plan is to eliminate the errors and bugs during the initial stage of the implementation. As the errors get debugged in the initial stage, the less complex the overall testing after integrating all the units of the system. The unit testing plan can be either simple or complex based on the functionality of that unit.

Testing Technique – Tool Selection Process

In this process the appropriate testing process is selected from various testing methodologies such as prototyping model, waterfall model etc and the selection is done by the means of analyzing the nature of the project. We go for Waterfall model.

Select test factor

This phase selects the appropriate test factor. The particular module of the project which is essential for the testing methodologies is sorted out first. This will help the testing process to be completed within time. The test factors for our project include encoding, verification and decoding process.

Determine SDLC phase

This phase involves the structural testing of the project which will be used for easy implementations of the functions. Though structural testing is so much associated with the coding phase, the structural testing should be carried out at all the phases of the lifecycle. These evaluates that all the structures are tested and sound.

Identify the criteria to test

In this phase the testing unit is trained with the necessary constraints and limit with which the project is to be tested. In our project the testing unit is trained to test whether the image to be encoded is in the PGM format.

Select type of test

Individual responsible for testing may prefer to select their own technique and tool based on the test situation. For selecting the appropriate testing process the project should be analyzed with the following three testing concepts:

- Structural versus functional testing
- Dynamic versus static testing
- Manual versus automatic testing

After analyzing through the above testing concepts we divided to test our project in Waterfall model testing methodology.

Structural Testing

Structural analysis based test sets are tend to uncover errors that occur during coding of the program. The properties of the test set are to reflect the internal structure of the program. Structural testing is designed to verify that the developed system and programs work as specified in the requirement. The objective is to ensure that the product is designed structurally sound and will function correctly.

Functional Testing

Functional testing ensures that the requirements are properly satisfied by the application system. The functions are those tasks that the system is designed to accomplish. This is not concerned with how processing occurs but rather with the results of the processing. The functional analysis based test sets tend to uncover errors that occurred in implementing requirements or design specifications.

Select technique

After selecting the appropriate testing methodology we have to select the necessary testing technique such as stress testing, execution testing, recovery testing, operation testing, compliance testing and security testing. We are performing operation testing

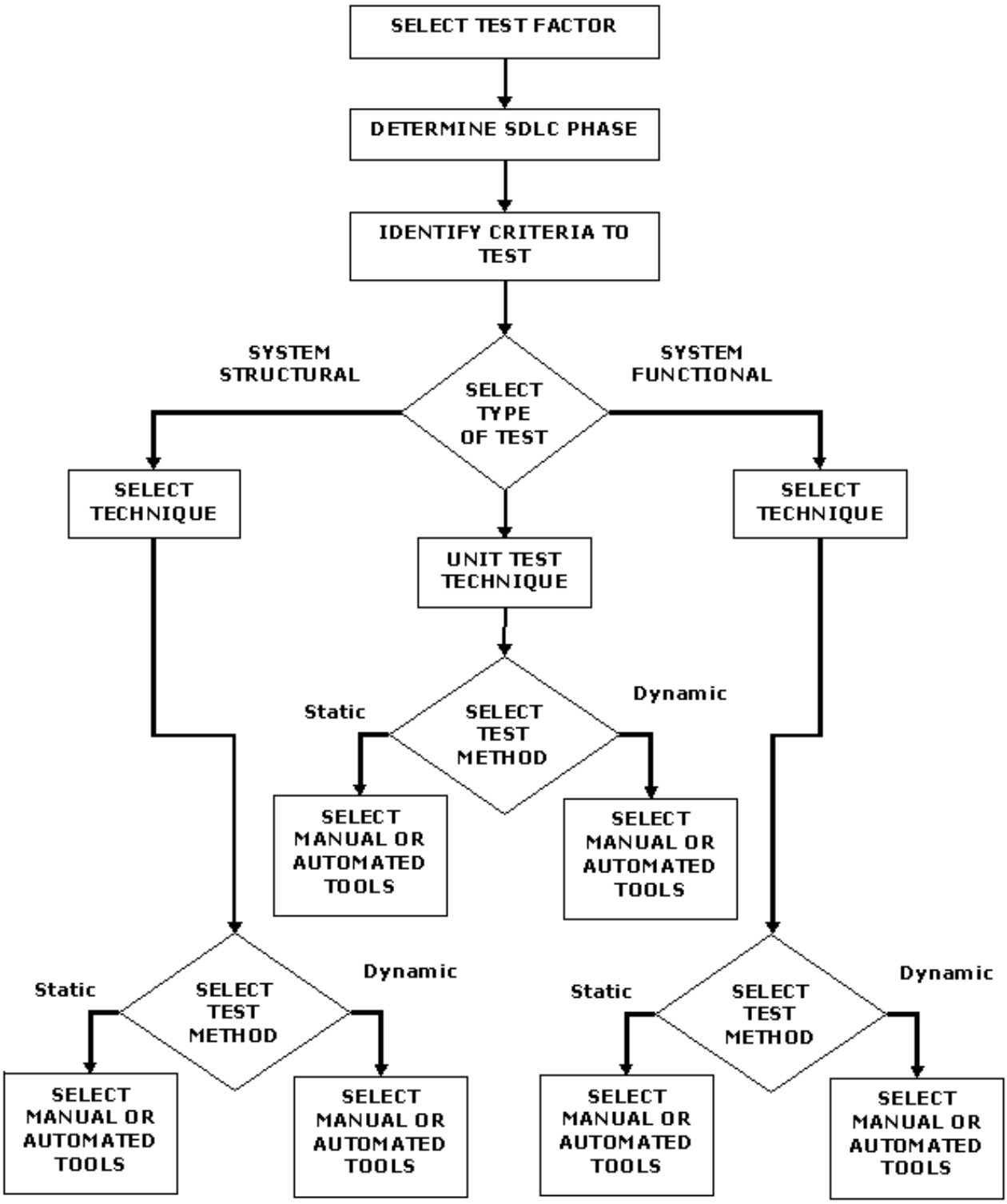


Fig 7.1 Testing technique and tool selection process

Select test method

We have to select the testing method which is to be carried out throughout the lifecycle. The two different methods are static and dynamic. Dynamic testing needs the program to be executed completely before testing. This is a traditional concept where faults

detected at the end will be very hard to rectify. In static process the program is tested for each and every line and the testing process is allowed to pass through only after rectifying the occurred fault. These make this process more expensive, so a combination of both static and dynamic testing method

Mode of testing

It is necessary to select the test mode in which the testing method to be carried out. The two different modes are manual and automated tool. The real time projects needs frequent interactions. So, it is impossible to carry out the testing process by means of automated tool. Our project uses manual testing.

Unit test technique

This phase examines the techniques, assessment and management of unit testing and analysis. Testing and analysis strategies are categorized according to whether they goal is functional or structural or combination of these. It will assist a software engineer to define, conduct and evaluate unit tests and to assess new unit test techniques.

System Testing

Once the entire system has been built then it has to be tested against the "System Specification" to check if it delivers the features required. It is still developer focused, although specialist developers known as systems testers are normally employed to do it. In essence System Testing is not about checking the individual parts of the design, but about checking the system as a whole. In effect it is one giant component. System testing can involve a number of specialist types of test to see if all the functional and non-functional requirements have been met.

Acceptance Testing

Acceptance Testing checks the system against the "Requirements". It is similar to systems testing in that the whole system is checked but the important difference is the change in focus. Systems Testing checks that the system that was specified has been delivered. Acceptance Testing checks that the system delivers what was requested. The customer, and not the developer should always do acceptance testing. The customer knows what is required from the system to achieve value in the business and is the only person qualified to makejudgment.

Regression Testing

This involves assurance that all aspects of an application system remain functional after testing. The introduction of change is the cause of problems in

previously tested segments. It is retesting unchanged segments of the application system. It normally involves rerunning tests that have been previously executed to ensure that the same results can be achieved currently as achieved when the segments were last tested.

7.3 Types of Testing:

Smoke Testing

Is the process of initial testing in which tester looks for the availability of all the functionality of the application in order to perform detailed testing on them. (Main check is for available forms)

Sanity Testing

Is a type of testing that is conducted on an application initially to check for the proper behavior of an application that is to check all the functionality are available before the detailed testing is conducted by on them.

Regression Testing

Is one of the best and important testing. Regression testing is the process in which the functionality, which is already tested before, is once again tested whenever some new change is added in order to check whether the existing functionality remains same.

Re-Testing

Is the process in which testing is performed on some functionality which is already tested before to make sure that the defects are reproducible and to rule out the environments issues if at all any defects are there.

Static Testing

Is the testing, which is performed on an application when it is not been executed. Ex: GUI, Document Testing

Dynamic Testing

Is the testing which is performed on an application when it is being executed. Ex: Functional testing.

Alpha Testing

It is a type of user acceptance testing, which is conducted on an application when it is just before released to the customer.

Beta-Testing

It is a type of UAT that is conducted on an application when it is released to the customer, when deployed in to the real time environment and being accessed by the real time users.

Monkey Testing

is the process in which abnormal operations, beyond capacity operations are done on the application to check the stability of it in spite of the users abnormal behavior.

Compatibility testing

It is the testing process in which usually the products are tested on the environments with different combinations of databases (application servers, browsers…etc) In order to check how far the product is compatible with all these environments platform combination.

Installation Testing

it is the process of testing in which the tester try to install or try to deploy the module into the corresponding environment by following the guidelines produced in the deployment document and check whether the installation is successful or not.

Adhoc Testing

Adhoc Testing is the process of testing in which unlike the formal testing where in test case document is used, with out that test case document testing can be done of an application, to cover that testing of the future which are not covered in that test case document. Also it is intended to perform GUI testing which may involve the cosmetic issues.

7.4 Test cases Table:

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

Test case 1

Summary: Invalid username or password.

Test Objective: To check whether the registered users are logging with their registered id’s or not.

Test Description: Only the registered users can login and can file download .

Test Environment: At the time of logging into the system.

Pre-Conditions: User details should be stored into the database.

Actions: User enters the registered userid and password.

Expected Results: “Successful login”.

Test case2

Summary: User download file

Test Objective: To check whether user is searching for a valid name that is in database.

Test Description: If the search data is not found the user can search again.

Test Environment: searching for alias data.

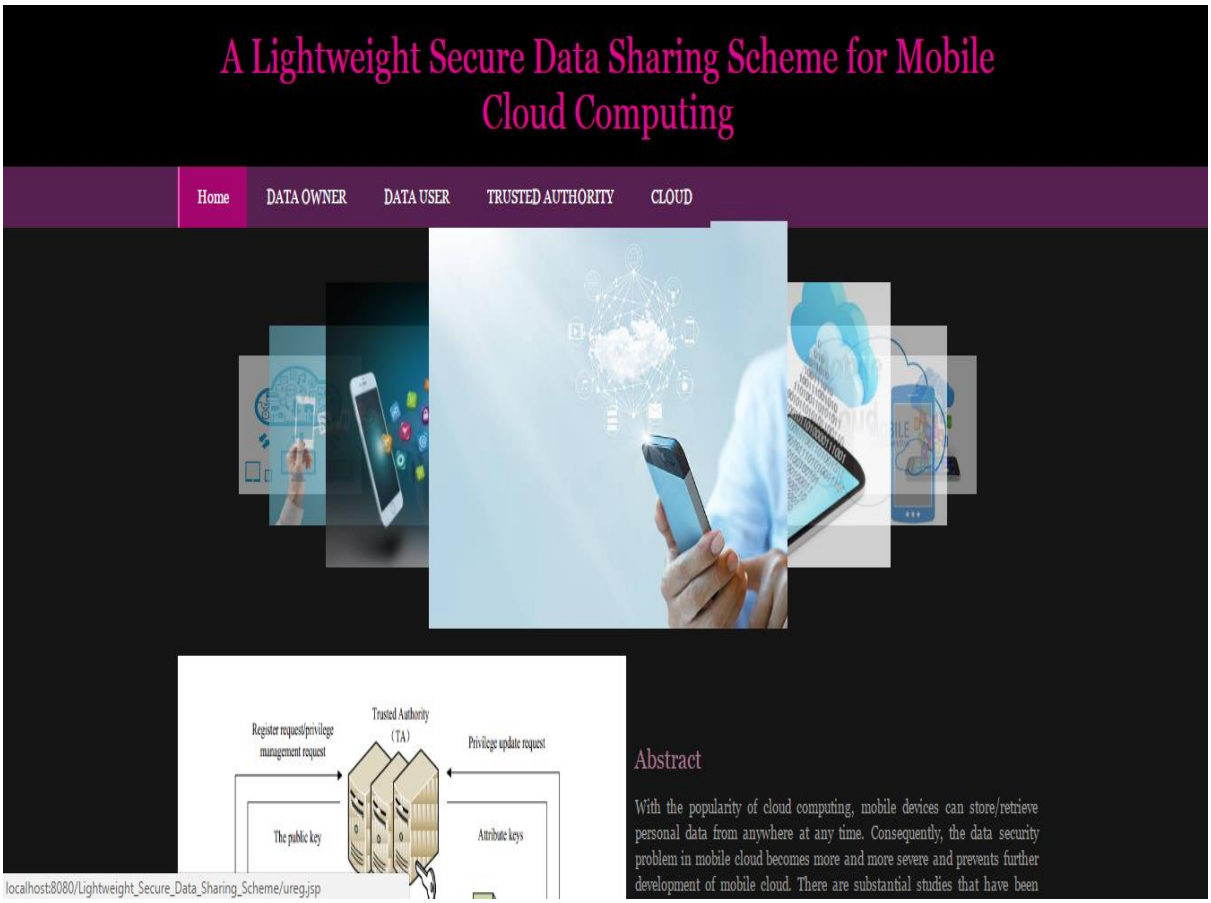
Pre-Conditions: All the alias names of a specified user should be stored in the database.

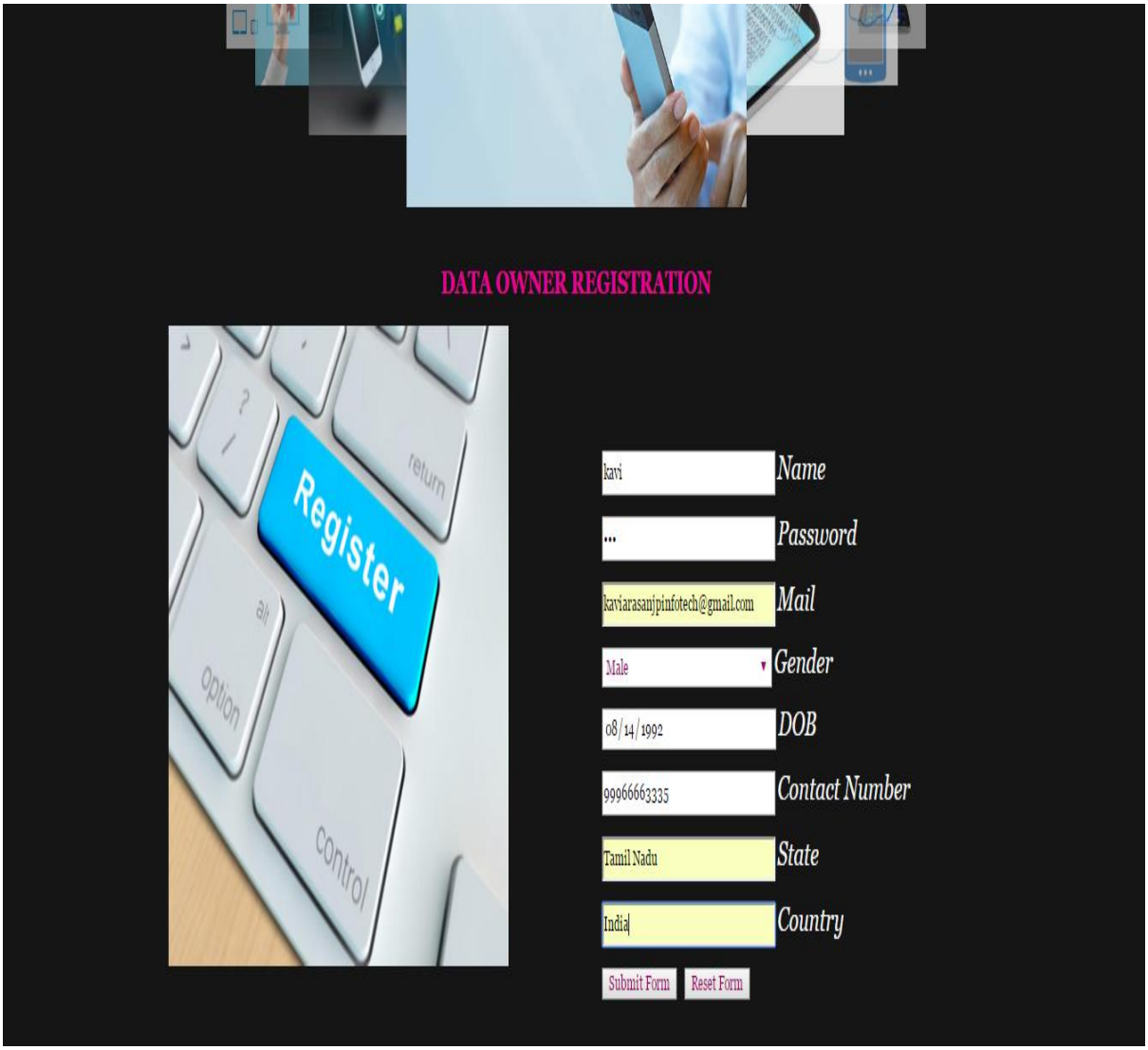
Actions: The user searches for a alias name for a personal name.

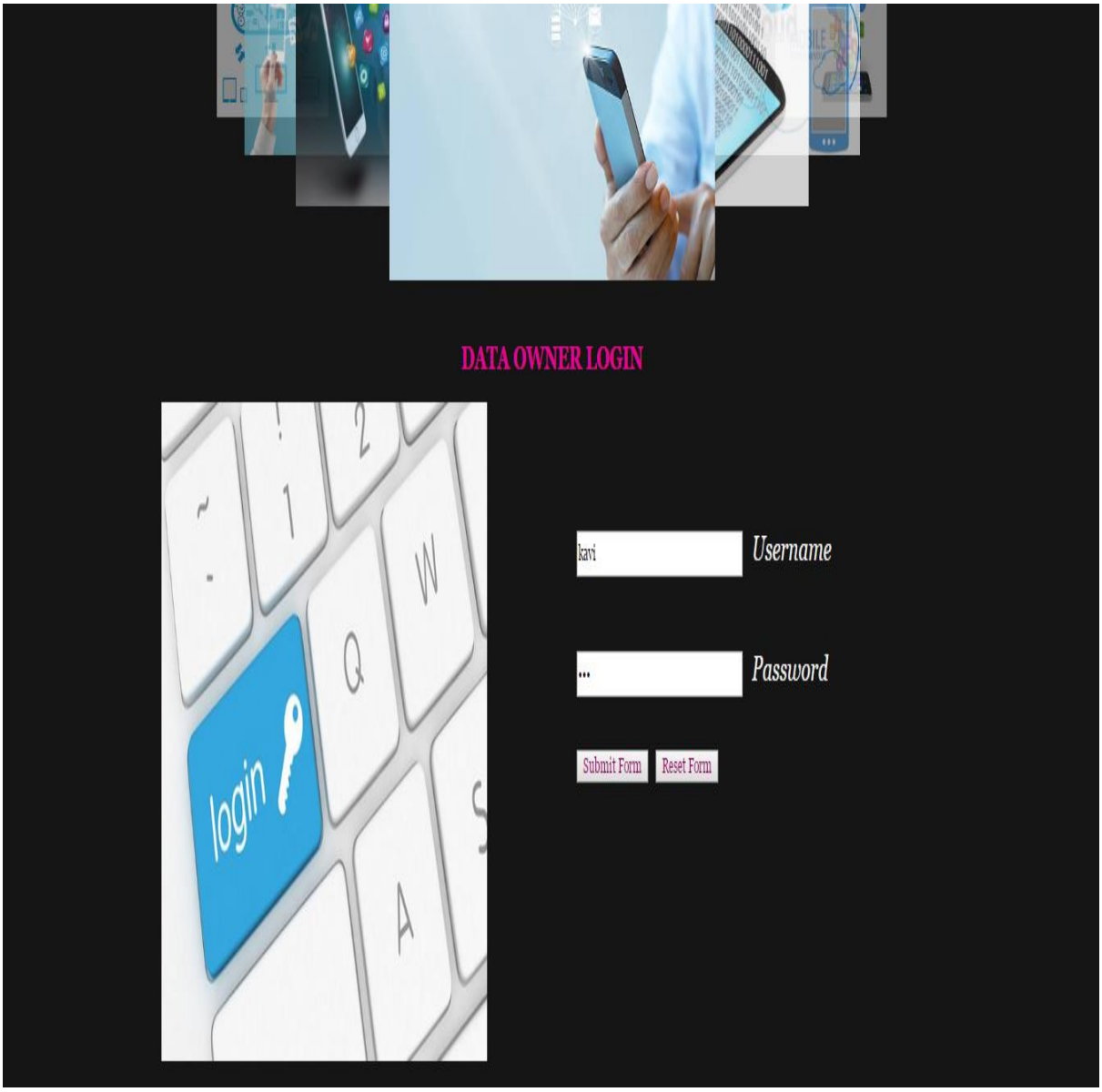
Expected Results: List of alias names are displayed.

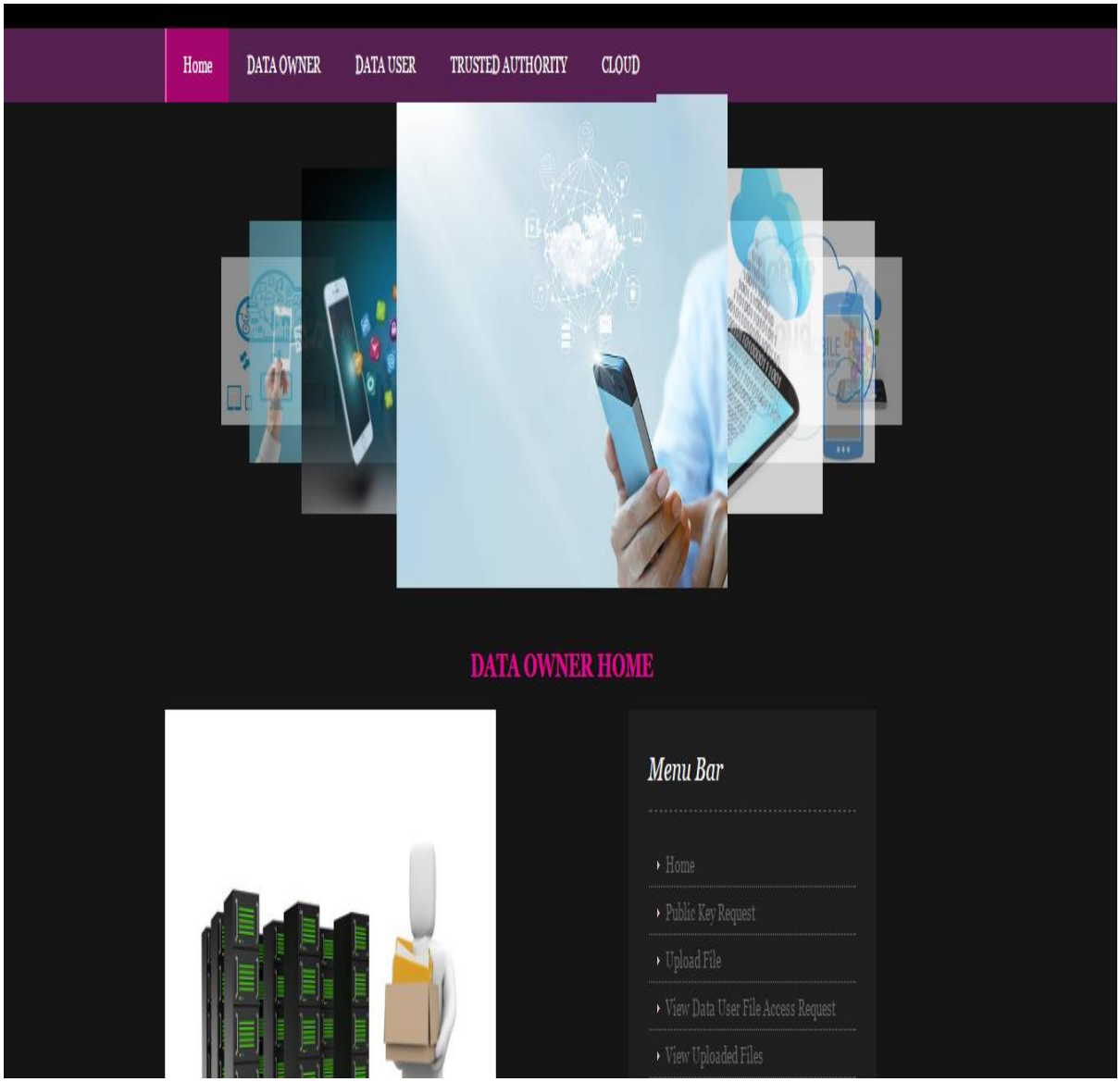
7. 5 Screen Shots:

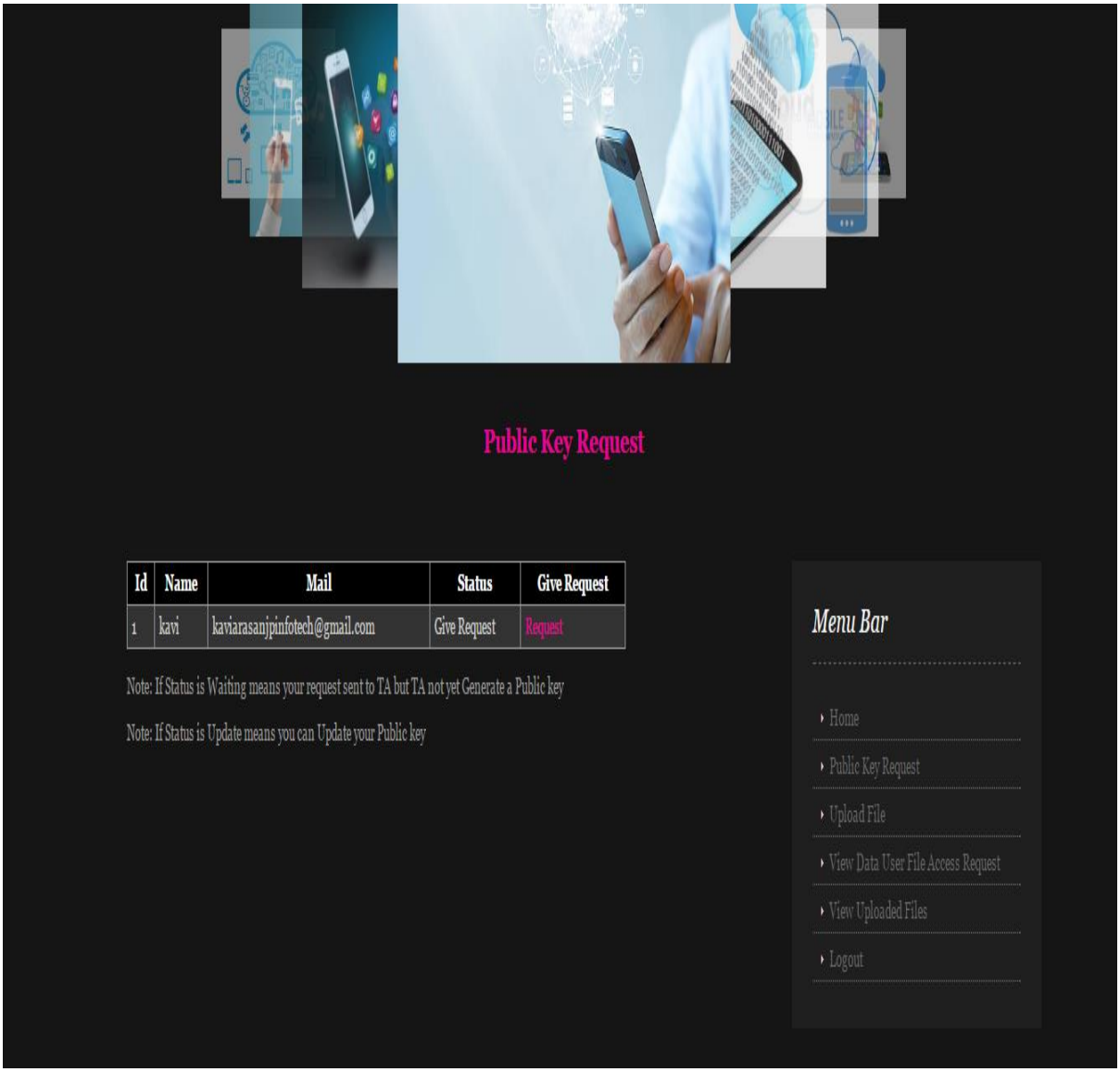
SCREEN SHOTS

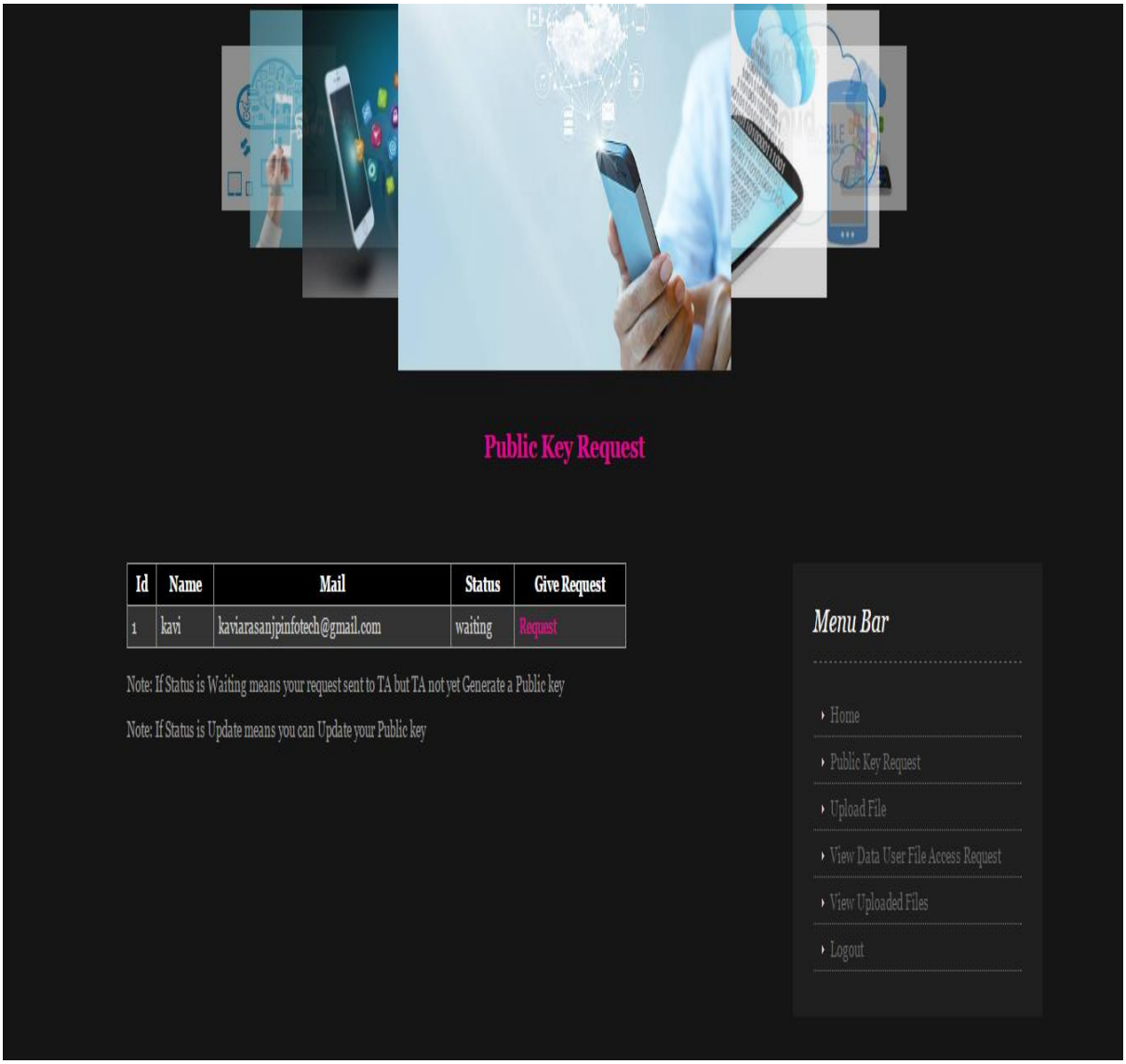



DATA OWNER REGISTRATION
Register
kavi
Name
...
Password
kaviarasanipinfotech@gmail.com
Mail
Male
Gender
08/14/1992
DOB
99966663335
Contact Number
Tamil Nadu
State
India
Country
Submit Form
Reset Form


DATA OWNER LOGIN
login
Kavi
Username
...
Password
Submit Form
Reset Form


Home
DATA OWNER
DATA USER
TRUSTED AUTHORITY
CLOUD
DATA OWNER HOME
Menu Bar
Home
Public Key Request
Upload File
View Data User File Access Request
View Uploaded Files


Public Key Request
Id
Name
Mail
Status
Give Request
1
kavi
kaviarasanjpinfotech@gmail.com
Give Request
Request
Note: If Status is Waiting means your request sent to TA but TA not yet Generate a Public key
Note: If Status is Update means you can Update your Public key
Menu Bar
Home
Public Key Request
Upload File
View Data User File Access Request
View Uploaded Files
Logout


Public Key Request
Id
Name
Mail
Status
Give Request
1
kavi
kaviarasanjpinfotech@gmail.com
waiting
Request
Note: If Status is Waiting means your request sent to TA but TA not yet Generate a Public key
Note: If Status is Update means you can Update your Public key
Menu Bar
Home
Public Key Request
Upload File
View Data User File Access Request
View Uploaded Files
Logout

8. CONCLUSION AND FUTURE ENHANCEMENTS

Conclusion:

In recent years, many studies on access control in cloud are based on attribute-based encryption algorithm (ABE). However, traditional ABE is not suitable for mobile cloud because it is computationally intensive and mobile devices only have limited resources. In this paper, we propose LDSS to address this issue. It introduces a novel LDSS-CP-ABE algorithm to migrate major computation overhead from mobile devices onto proxy servers, thus it can solve the secure data sharing problem in mobile cloud. The experimental results show that LDSS can ensure data privacy in mobile cloud and reduce the overhead on users' side in mobile cloud. In the future work, we will design new approaches to ensure data integrity. To further tap the potential of mobile cloud, we will also study how to do ciphertext retrieval over existing data sharing schemes.

Future Enhancement

Future work could include extending the algorithm to work with multiple vertex attributes, multiple edges and edge attributes, and wildcard queries. Because dual simulation is often the bottleneck of the algorithm, ways to parallelize or speed up the dual simulation algorithm would bring further improvements. Adapting the algorithm to work on graphs stored on disk or in a distributed environment would allow it to handle even larger graphs.

9. BIBLIOGRAPHY

[1] Gentry C, Halevi S. Implementing gentry's fully-homomorphic encryption scheme. in: Advances in Cryptology–EUROCRYPT 2011. Berlin, Heidelberg: Springer press, pp. 129-148, 2011.

[2] Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE. in: Proceeding of IEEE Symposium on Foundations of Computer Science. California, USA: IEEE press, pp. 97-106, Oct. 2011.

[3] Qihua Wang, Hongxia Jin. "Data leakage mitigation for discertionary access control in collaboration clouds". the 16th ACM Symposium on Access Control Models and Technologies (SACMAT), pp.103-122, Jun. 2011.

[4] Adam Skillen and Mohammad Mannan. On Implementing Deniable Storage Encryption for Mobile Devices. the 20th Annual Network and Distributed System Security Symposium (NDSS), Feb. 2013.

[5] Wang W, Li Z, Owens R, et al. Secure and efficient access to outsourced data. in: Proceedings of the 2009 ACM workshop on Cloud computing security. Chicago, USA: ACM pp. 55-66, 2009.

[6] Maheshwari U, Vingralek R, Shapiro W. How to build a trusted database system on untrusted storage. in: Proceedings of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4. USENIX Association, pp. 10-12, 2000.

[7] Kan Yang, Xiaohua Jia, Kui Ren: Attribute-based fine-grained access control with efficient revocation in cloud storage systems. ASIACCS 2013, pp. 523-528, 2013.

[8] Crampton J, Martin K, Wild P. On key assignment for hierarchical access control. in: Computer Security Foundations Workshop. IEEE press, pp. 14-111, 2006.

[9] Shi E, Bethencourt J, Chan T H H, et al. Multi-dimensional range query over encrypted data. in: Proceedings of Symposium on Security and Privacy (SP), IEEE press, 2007. 350- 364

[10] Cong Wang, Kui Ren, Shucheng Yu, and Karthik Mahendra Raje Urs. Achieving Usable and Privacy-assured Similarity Search over Outsourced Cloud Data. IEEE INFOCOM 2012, Orlando, Florida, March 25-30, 2012

[11] Yu S., Wang C., Ren K., Lou W. Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. INFOCOM 2010, pp. 534-542, 2010

[12] Kan Yang, Xiaohua Jia, Kui Ren, Bo Zhang, Ruitao Xie: DACMACS: Effective Data Access Control for Multiauthority Cloud Storage Systems. IEEE Transactions on Information Forensics and Security, Vol. 8, No. 11, pp.1790-1801, 2013.

[13] Stehlé D, Steinfeld R. Faster fully homomorphic encryption. in: Proceedings of 16th International Conference on the Theory and Application of Cryptology and Information Security. Singapore: Springer press, pp.377-394, 2010.

[14] Junzuo Lai, Robert H. Deng ,Yingjiu Li ,et al. Fully secure keypolicy attribute-based encryption with constant-size ciphertexts and fast decryption. In: Proceedings of the 9th ACM symposium on Information, Computer and Communications Security (ASIACCS), pp. 239-248, Jun. 2014.

[15] Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute based encryption in: Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP). Washington, USA: IEEE Computer Society, pp. 321-334, 2007.

[16] Liang Xiaohui, Cao Zhenfu, Lin Huang, et al. Attribute based proxy re-encryption with delegating capabilities. in: Proceedings of the 4th International Symposium on Information, Computer and Communications Security. New
York, NY, USA: ACM press, pp. 276-286, 2009.

[17] Pirretti M, Traynor P, McDaniel P, et al. Secure atrribute-based systems. in: Proceedings of the 13th ACM Conference on Computer and Communications Security. New York, USA: ACM press, pp. 99-112, 2006.

[18] Yu S., Wang C., Ren K., et al. Attribute based data sharing with attribute revocation. in: Proceedings of the 5th International Symposium on Information, Computer and Communications Security (ASIACCS), New York, USA: ACM press pp. 261-270, 2010.

[19] Sandhu R S, Coyne E J, Feinstein H L, et al. Role-based access control models. Computer, 29(2): 38-47, 1996.

[20] Tian X X, Wang X L, Zhou A Y. DSP RE-Encryption: A flexible mechanism for access control enforcement management in DaaS. in: Proceedings of IEEE International Conference on Cloud Computing. IEEE press, pp.25-32, 2009

[21] Di Vimercati S D C, Foresti S, Jajodia S, et al. Over-encryption: management of access control evolution on outsourced data. in: Proceedings of the 33rd international conference on Very large data bases. Vienna, Austria: ACM, pp. 123-134, 2007.

[22] Kan Yang, Xiaohua Jia, Kui Ren, Ruitao Xie, Liusheng Huang: Enabling efficient access control with dynamic policy updating for big data in the cloud. INFOCOM 2014, pp.2013-2021, 2014.

[23] Jia W, Zhu H, Cao Z, et al. SDSM: a secure data service mechanism in mobile cloud computing. in: Proceedings of 30th IEEE International Conference on Computer Communications. Shanghai, China: IEEE, pp. 1060-1065, 2011.

[24] D. Huang, X. Zhang, M. Kang, and J. Luo. Mobicloud: A secure mobile cloud framework for pervasive mobile computing and communication. in: Proceedings of 5th IEEE International Symposium on Service-Oriented System Engineering. Nanjing, China: IEEE, pp. 90-98, 2010.

[25] Benjamin Livshits, Jaeyeon Jung. Automatic Mediation of Privacy-Sensitive Resource Access in Smartphone Applications. USENIX Security, pp.113-130, Aug. 2013.

[26] Zhou Z, Huang D. Efficient and secure data storage operations for mobile cloud computing. in: Proceedings of 8th International Conference on Network and Service Management (CNSM 2012), Las Vegas, USA: IEEE, pp. 37-45, 2012.

[27] P. K. Tysowski and M. A.Hasan. Hybrid attribute- and reencryption-based key management for secure and scalable mobile applications in clouds. IEEE Transactions on Cloud Computing, vol. 1, no. 2, pp. 172-186, Nov. 2013.

[28] Boneh D, Franklin M. Identity-based encryption from the Weil pairing. in: Proceedings of the Advances in Cryptology. Berlin, Heidelberg: Springer-Verlag, pp. 213−229, 2001.

[29] Sahai A, Waters B. Fuzzy identity based encryption. in: Proceedings of the Advances in Cryptology. Aarhus, Denmark: Springer-Verlag, pp.457-473, 2005.

[30] Shamir A. How to share a secret. Communications of the ACM,1979, 22 (11): 612-613.
87–1706, 2013.