A

Report on

<<Title of the project>>

Thesis submitted in partial fulfillment of the requirement for the award of the degree of

**Master of Technology**

**In**

**<<Branch Name>>**

Submitted By

<< Name of the Student >>

<< Hall Ticket No >>

*Under the Esteemed Guidance of*

<< Guide Name >>

<< Designation >>

<< College Logo >>

Department of << Department name >>,

<< College name (Upper case only) >>

(Affiliated to Jawaharlal Nehru Technological University)

<<Distinct Name >>-Pin code, << State name>> INDIA

2015

**<< College Name >>**

**(Affiliated to Jawaharlal Nehru Technological University)**

**<< Distinct Name >>- << Pin code >>. << State name >>., INDIA**

**CERTIFICATE**

**<< College Logo >>**

This is to certify that the Thesis entitled “**<<**title of the project**>>”** is being submitted by << student name>>bearing **<<**Roll number**>>** in partial fulfilment of the requirement for the award of the degree of M.Tech in << **Branch Name >>** to the <<College Name>>, <<University Name >> is a record of bonafide work carried out by him/her under my guidance. The results presented in this thesis have been verified and are found to be satisfactory. The results embodied in this thesis have not been submitted to any other University for the award of any other degree or diploma.

**Project Guide**  **Head of the Department**

**<<Guide Name>>, <<HOD Name>>**

**<<Designation>> <<Designation>>**

**EXTERNAL PRINCIPAL**

**Next page:** This page is required for the candidates who pursue the project work outside (other

organization), under the supervision of an **EXTERNAL GUIDE.** This page is another certificate

given on the organization letterhead where the project work is pursued, which should be certified

and duly signed by the External guide about work done by the candidate and clearly mentioning the duration of the project work. The format for this page will almost be the same as the previous page with double spacing between the lines.

**(or)**

Include the Declaration page in the below specified Format:

**DECLARATION**

I **<<Student Name>>** bearing **Roll Number <<Roll No>>**, a bonafide student of <<**College Name >>**, would like to declare that the project titled **“<<project title>>”** a partial fulfillment of **M.Tech** Degree course of << **University Name >>** is my original work in the year 2015 under the guidance of **<< Guide Name>>,** <<**Designation>>** of the **<< Department Name >> Department**.

Date: **<<Student Name>>**

**(Roll Number)**

**Acknowledgement**

I would like to express my sincere gratitude to my advisor, **<<Guide Name>>**, whose knowledge and guidance has motivated me to achieve goals I never thought possible. He has consistently been a source of motivation, encouragement, and inspiration. The time I have spent working under his supervision has truly been a pleasure.

I thank H.O.D **<<HOD Name>>** for his effort and guidance and all senior faculty members of CSE Department for their help during my course. Thanks to programmers and non-teaching staff of C.S.E Department of VITS.

I Thank my principal **<<PRINCIPAL NAME>>** and Management for providing excellent facilities to carry out my project work.

Finally Special thanks to my parents for their support and encouragement throughout my life and this course. Thanks to all my friends and well wishers for their constant support.

**Names of Student (roll number)**

**Seventh page:** The seventh page may contain an **ABSTRACT** of the Thesis. The candidates may emphasize here his/her contributions.

**Page 8th, 9th …..:** In these pages candidate must provide a **TABLE OF CONTENTS**, list of tables and list of figures, Screens and notation.

**NOTE:** All the above pages are to be numbered in Roman numerals of lower case.

**Arrangement of chapters:** The following is the suggested format for arranging the Thesis matter into various chapters:

**1) Introduction** ( Do not copy as it ease from Base Paper)

**2) Literature Survey** ( Quantity should be 10 – 15 Pages)

**3) SRS** ( Preferably IEEE Format- Do not include Technology Desciption))

**4) Analysis and Design** (Preferably ER-Diagrams & DFD)

**5) Implementation** ( Architecture, Algorithms used, Module detailed desc)

**6) Testing** (Min. 10 test caes)

**7) Results** (Navigation should be provided)

**8) Conclusion and Future Scope** (In two paragraphs)

**9) Reference / Bibliography** (Format is given below)

**10) Appendices** (if any)

a) Standards (if any)

b) Sample Code(Optional)

**The arrangement of paragraph in a chapter:** Each paragraph in a chapter should be properly numbered for example: 2.1, 2.2, etc., where the first digit represents the chapter number and the second digit the paragraph number. There is no need to indicate the number for the first paragraph in a chapter.

Sub-paragraphs, if any, may be indicated as 1.1.1, 1.1.2, etc., i.e., the first digit representing the chapter, the second representing the paragraph and the third representing the sub-paragraph.

**Photographs and Tables:** The Figures and tables occurring in a chapter may be serially numbered as **Fig: 1.1, 1.2,** etc., where the first digit represents the chapter, the second digit represents the figure number. ( It should be specified below the figure)

The tables may be represented as **Tab 1.1, 1.2,** etc., where the first digit represents the chapter,

the second digit represents the table number. ( It should be specified above the Table)

**Graphs:** The graph should clearly indicate the pinots which are used for drawing the curve or

curves. All the letters in the graph should be written with stencils.

**Bibliography or References:** The following format may be used for writing the Bibliography /References.

` At the end of Thesis where the listing of references is done, the list should be made strictly in alphabetic order of the names of the author. The references/websites have to be listed in the following format:

**[S.No] Author, Paper/Book, Publisher/Magazine/Conference, Volume/Edition, page numbers,Year**

**Examples:**

[1] Bruce, Cryptography, Tata McGraw Hill, 1978

[2] R. R. Duncan, "Remediation of Lead in Water Supplies," IEEE Trans. Microwave Theory Tech.,vol. 99, no. 18, pp. 257-278, Nov. 1986.

[3] <http://www.google.com>

**Guidelines for preparing the Documentation for M.Tech.**

**Size and copies:** The Documentation should be submitted in A4 size and 5 copies (4 Hard copies + 1Soft copy – CD) of the Thesis.

**Paper, Typing and Format:** Bond paper should be used for the preparation of the Thesis. Typing should be done on the 12 point size letters for the running text, 14 point size for the sub-headings and 16 point size for main headings/titles/chapter names/etc. The font type should be preferably TIMES NEW ROMAN.

The layout margin to be provided is 1.5’ on the left, 1’ on the top and bottom, 1’ on the right.

Fresh paragraph should commence after tab space. 1.5 line spacing shall be provided throughout the Thesis.

The page number shall be indicated at the bottom right of the each page. Front Pages Small Roman Numbers (Excluding title page, Certificate page, Acknowledgement page), Body pages 1,2,3 ………., Annexure 1,2,3……….. (Separate for each Annexure).

**Binding**: The Thesis shall be properly bound using resin of **Cement gray color** (Cement Color) for M.Tech. The bound front cover should indicate in golden embossed letter the following:

**Guidelines for the project work of all PG courses:**

1. **Nature of the content**: The project work must be an evidence of the utilization and practical application of the knowledge / techniques learnt in the theory courses pertaining to the faculty / speciality.

2. **Quality and quantity:** Both the extent and the level / standard of the work carried out by the student must be greater than that of an undergraduate project and be commensurate with the depth of coverage and sophistication reflected in the theory courses of the Post Graduate specialization.

3. **Originality:** A mere collection and review, how comprehensive it may be of the previous works or the state-of-art cannot qualify as the project work, though it can form a part of it.

4. The project work must be the result of the theoretical and / or practical investigation actually carried out by the candidate.

5. It must reflect the originality of the student and mere duplication of earlier works does not suffice. However, extensions and modifications of earlier work(s) are acceptable if there is original contribution by the candidate.

6. It should be possible to publish at least one paper in a reviewed journal from the work presented in the thesis.

\* \* \*

**A Lightweight Secure Data Sharing Scheme for Mobile Cloud Computing**

**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 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. To address this issue, in this paper, we propose a Lightweight Data Sharing Scheme (LDSS) for mobile cloud computing environment. The main contributions of LDSS are as follows: (1) We design an algorithm called LDSS-CP-ABE based on Attribute-Based Encryption (ABE) method to offer efficient access control over ciphertext. (2) 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. (3) We introduce lazy re-encryption and description field of attributes to reduce the revocation overhead when dealing with the user revocation problem. (4) Finally, we implement a data sharing prototype framework based on LDSS. 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.

**Objective of the Project**

With the popularity of cloud computing, mobile devices can store/retrieve personal data from anywhere at any time. Consequently, the data security problem in mobile cloud becomes more and more severe and prevents further development of mobile cloud. There are substantial studies that have been conducted to improve the cloud security. However, most of them are not applicable for mobile cloud since mobile devices only have limited computing resources and power. Solutions with low computational overhead are in great need for mobile cloud applications. In this paper, we propose a lightweight data sharing scheme (LDSS) for mobile cloud computing. It adopts CP-ABE, an access control technology used in normal cloud environment, but changes the structure of access control tree to make it suitable for mobile cloud environments. LDSS moves a large portion of the computational intensive access control tree transformation in CP-ABE from mobile devices to external proxy servers. Furthermore, to reduce the user revocation cost, it introduces attribute description fields to implement lazy-revocation, which is a thorny issue in program based CP-ABE systems. The experimental results show that LDSS can effectively reduce the overhead on the mobile device side when users are sharing data in mobile cloud environments.

**2. LITERATURE SURVEY**

**Review of access control models for cloud computing**

The relationship between users and resources is dynamic in the cloud, and service providers and users are typically not in the same security domain. Identity-based security (e.g., discretionary or mandatory access control models) cannot be used in an open cloud computing environment, where each resource node may not be familiar, or even do not know each other. Users are normally identified by their attributes or characteristics and not by predefined identities. There is often a need for a dynamic access control mechanism to achieve crossdomain authentication. In this paper, we will focus on the following three broad categories of access control models for cloud computing: (1) Role-based models; (2) Attribute-based encryption models and (3) Multi-tenancy models. We will review the existing literature on each of the above access control models and their variants (technical approaches, characteristics, applicability, pros and cons), and identify future research directions for developing access control models for cloud computing environments.

We identify the following future research directions for access control models in cloud computing environments: (1) Develop attribute-driven role-based access control models such that the user role and role-permission assignments be separately constructed using policies applied on the attributes of users, roles, the objects and the environment; and the attribute-based user-role and role-permission assignment rules be applied in real-time to enforce access control decisions. (2) Develop a location-aware role-based control model incorporated to the Policy Enforcement Point of a cloud (thereby, preventing the disclosure of user’s identity, role, or location directly to a remote server in the cloud that may not be fully trusted), and enable/activate the role only when the user is located within the logical positions (computed from real positions by specific mapping functions) that lie within the spatial boundary of a role. (3) Explore software-hardware co-design for security such that the fine-grained access control and usage control mechanisms implemented in software are integrated with new hardware architectural and virtualization features that can help protect the confidentiality and integrity of the data and the resources, even when the powerful underlying hypervisor may be compromised. (4) Mitigate insider threats to the data and resources from the perspective of both a rogue cloud provider administrator and the employee in the victim organization that exploits cloud weaknesses for unauthorized access. (5) Incorporate the relationship between trust and reputation in the access control models for better and secure service quality within the cloud.

**On Implementing Deniable Storage Encryption for Mobile Devices**

Data confidentiality can be effectively preserved through encryption. In certain situations, this is inadequate, as users may be coerced into disclosing their decryption keys. In this case, the data must be hidden so that its very existence can be denied. Steganographic techniques and deniable encryption algorithms have been devised to address this specific problem. Given the recent proliferation of smartphones and tablets, we examine the feasibility and ef- ficacy of deniable storage encryption for mobile devices. We evaluate existing, and discover new, challenges that can compromise plausibly deniable encryption (PDE) in a mobile environment. To address these obstacles, we design a system called Mobiflage that enables PDE on mobile devices by hiding encrypted volumes within random data on a device’s external storage. We leverage lessons learned from known issues in deniable encryption in the desktop environment, and design new countermeasures for threats specific to mobile systems. Key features of Mobiflage include: deniable file systems with limited impact on throughput; efficient storage use with no data expansion; and restriction/prevention of known sources of leakage and disclosure. We provide a proof-of-concept implementation for the Android OS to assess the feasibility and performance of Mobiflage. We also compile a list of best practices users should follow to restrict other known forms of leakage and collusion that may compromise deniability.

Mobile devices are increasingly being used for capturing and spreading images of popular uprisings and civil disobedience. To keep such records hidden from authorities, deniable storage encryption may offer a viable technical solution. Such PDE-enabled storage systems exist for mainstream desktop/laptop operating systems. With Mobi- flage, we explore design and implementation challenges of PDE for mobile devices, which may be more useful to regular users and human rights activists. Mobiflage’s design is partly based on the lessons learned from known attacks and weaknesses of desktop PDE solutions. We also consider unique challenges in the mobile environment (such as ISP or wireless carrier collusion with the adversary). To address some of these challenges, we need the user to comply with certain requirements. We compiled a list of rules the user must follow to prevent leakage of information that may weaken deniability. Even if users follow all these guidelines, we do not claim that Mobiflage’s design is completely safe against any leaks (cf. [10]). We want to avoid giving any false sense of security. We present Mobiflage here to encourage further investigation of PDE-enabled mobile systems. Source code of our prototype implementation is available on request.

**Secure and Efficient Access to Outsourced Data**

Providing secure and efficient access to large scale outsourced data is an important component of cloud computing. In this paper, we propose a mechanism to solve this problem in owner-write-users-read applications. We propose to encrypt every data block with a different key so that flexible cryptography-based access control can be achieved. Through the adoption of key derivation methods, the owner needs to maintain only a few secrets. Analysis shows that the key derivation procedure using hash functions will introduce very limited computation overhead. We propose to use over-encryption and/or lazy revocation to prevent revoked users from getting access to updated data blocks. We design mechanisms to handle both updates to outsourced data and changes in user access rights. We investigate the overhead and safety of the proposed approach, and study mechanisms to improve data access efficiency.

we propose a mechanism to achieve secure and efficient access to outsourced data in owner-write-usersread applications. We assume that the outsourced data has a very large scale and we try to reduce the overhead at the data owner and service provider. We propose to encrypt every data block with a different key so that flexible cryptographybased access control can be achieved. Through the adoption of key derivation method, the owner needs to maintain only a few secrets. Analysis shows that the key derivation procedure based on hash functions will introduce very limited overhead. We propose to use over-encryption and/or lazy revocation to prevent revoked users from getting access to updated data blocks. We design mechanisms to handle both updates to outsourced data and changes in user access rights. We analyze the computational, storage, and communication overhead of the approach. We also investigate the scalability and safety of the approach. Extensions to our approach include the following aspects. First, we plan to design a new scheme for key management based on this approach so that it can be applied to manywrite-many-read applications. Second, we want to design dynamic mapping functions among keys in the hierarchy and index numbers of data blocks so that we can progressively reorganize the data blocks based on their access patterns. In this way, we can further reduce the number of keys that the owner sends to the end user. Finally, we plan to integrate existing approaches to access control, provable data possession, and key management for outsourced data to develop a new approach to secure Storage-as-a-Service.

**How to Build a Trusted Database System on Untrusted Storage**

Some emerging applications require programs to maintain sensitive state on untrusted hosts. This paper presents the architecture and implementation of a trusted database system, TDB, which leverages a small amount of trusted storage to protect a scalable amount of untrusted storage. The database is encrypted and validated against a collision-resistant hash kept in trusted storage, so untrusted programs cannot read the database or modify it undetectably. TDB integrates encryption and hashing with a low-level data model, which protects data and metadata uniformly, unlike systems built on top of a conventional database system. The implementation exploits synergies between hashing and log-structured storage. Preliminary performance results show that TDB outperforms an off-the-shelf embedded database system, thus supporting the suitability of the TDB architecture.

We have presented a trusted database system that leverages a trusted processing environment and a small amount of trusted storage to extend tamper-detection and secrecy to a scalable amount of untrusted storage. The architecture integrates encryption and hashing with a low-level data model, which protects data and metadata uniformly. The model is powerful enough to support higher-level database functions such as transactions, backups, and indexing. We found that log-structured storage is well suited for building such a system. The implementation is simplified by embedding a hash tree in the comprehensive location map that is central to log-structured systems: objects can be validated as they are located. The checkpointing optimization defers and consolidates the propagation of hash values up the tree. Because updates are not made in place, a snapshot of the database state can be created using copy-on-write, which facilitates incremental backups. We measured the performance of TDB using microbenchmarks as well as a high-level workload. The database overhead was dominated by writes to the untrusted store and the tamper-resistant store, which may vary significantly based on the types of devices used. The overhead of encryption and hashing was only 6% of the total. On this workload, TDB outperformed a system that layers cryptography on an off-the-shelf embedded database system, while also providing more protection. This supports the suitability of the TDB architecture.

**Attribute Based DRM Scheme with Efficient Revocation in Cloud Computing**

The existing digital rights management (DRM) schemes in cloud computing introduce a heavy computation overhead on the content provider for key distribution. In this paper, we propose an attribute-based DRM scheme in cloud computing by combining the techniques of ciphertextpolicy attribute-based encryption (CP-ABE) and proxy reencryption (PRE). We first divide the content encryption key into two parts, content master key and assistant key. Then we enforce access policies based on attributes to distribute the content master key securely. Thus the users who satisfy the access policy can recover the content master key, and then obtain assistant key from the key server and decrypt the content. Furthermore, we achieve efficient attribute and user revocation by allowing the attribute authority to delegate the key server to refuse to issue the assistant key for the revoked users. The security and performance analyses indicate that the proposed scheme is secure, efficient, and privacy-preserving.

Cloud computing is a new revolution in IT, and has the potential to reshape the business model of the IT industry. In this paper, we have proposed an attribute-based DRM scheme by combining the techniques of ciphertext-policy attribute-based encryption and proxy re-encryption. In the proposed scheme, we define and enforce access policy based on attributes to achieve fine-grained access control and privacy preserving. The users who satisfy the access policy can recover the content master key and obtain the assistant key from the key server and decrypt the content. Furthermore, we realize efficient user and attribute revocation by allowing the attribute authority to delegate most of the revocation tasks to key server without disclosing the assistant key. Compared with other DRM schemes in cloud computing, our scheme is highly efficient and provably secure. Our future work will be focused on dynamic usage control in cloud computing.

**A Novel Key Management Scheme for Dynamic Access Control in a Hierarchy**

Shen and Chen proposed a novel key management scheme for dynamic access control in a hierarchy. In this article, the authors shall present an improved version of Shen and Chen’s scheme to reduce the computational time required for key generation and derivation.

We have presented a revised scheme which is a slight modification of the Shen-Chen scheme to reduce the computational time. The improved scheme can also perform dynamical access control the same way the ShenChen scheme does.

**Multi-Dimensional Range Query over Encrypted Data**

We design an encryption scheme called Multi-dimensional Range Query over Encrypted Data (MRQED), to address the privacy concerns related to the sharing of network audit logs and various other applications. Our scheme allows a network gateway to encrypt summaries of network flows before submitting them to an untrusted repository. When network intrusions are suspected, an authority can release a key to an auditor, allowing the auditor to decrypt flows whose attributes (e.g., source and destination addresses, port numbers, etc.) fall within specific ranges. However, the privacy of all irrelevant flows are still preserved. We formally define the security for MRQED and prove the security of our construction under the decision bilinear Diffie-Hellman and decision linear assumptions in certain bilinear groups. We study the practical performance of our construction in the context of network audit logs. Apart from network audit logs, our scheme also has interesting applications for financial audit logs, medical privacy, untrusted remote storage, etc. In particular, we show that MRQED implies a solution to its dual problem, which enables investors to trade stocks through a broker in a privacy-preserving manner.

We design an encryption scheme that allows us to encrypt an arbitrary message and a set of attributes. An authority holding a master key can issue a search capability to an authorized party, allowing it to decrypt data entries whose attributes fall within specific ranges; while the privacy of other data entries is preserved. We prove the security of our scheme under the D-BDH and the D-Linear assumptions in certain bilinear groups. We also study the practical performance of our construction in network audit log applications. Apart from network audit logs, MRQED can be useful in various other applications such as financial audit logs, untrusted email servers and medical privacy. In particular, we show that the dual problem can be useful for investors who wish to trade stocks through a broker in a privacy-preserving manner.

**Privacy-Preserving Multikeyword Similarity Search Over Outsourced Cloud Data**

The amount of data generated by individuals and enterprises is rapidly increasing. With the emerging cloud computing paradigm, the data and corresponding complex management tasks can be outsourced to the cloud for the management flexibility and cost savings. Unfortunately, as the data could be sensitive, the direct data outsourcing would have the problem of privacy leakage. The encryption can be used, before the data outsourcing, with the concern that the operations can still be accomplished by the cloud. We consider the multikeyword similarity search over outsourced cloud data. In particular, with the consideration of the text data only, multiple keywords are specified by the user. The cloud returns the files containing more than a threshold number of input keywords or similar keywords, where the similarity here is defined according to the edit distance metric. We propose three solutions, where blind signature provides the user access privacy, and a novel use of Bloom filter’s bit pattern provides the speedup of search task at the cloud side. Our final design to achieve the search is secure against insider threats and efficient in terms of the search time at the cloud side. Performance evaluation and analysis are used to demonstrate the practicality of our proposed solutions.

We consider the problem of PPMKSS over outsourced cloud data, for the first time in the literature. With the keyword suppressing technique and the Bloom filter, three solutions, namely, PPMKSS-1, PPMKSS-2, and PPMKSS-3, are proposed as candidates for dealing with such search problem. In particular, PPMKSS-3 is highly efficient in terms of storage, computation, and communication overhead. Moreover, we also design a user authorization mechanism based on blind signature, to ensure the user access privacy. As a whole, based on our evaluation, the proposed schemes can be practically useful in offering PPMKSS.

**Achieving sheltered, scalable and fine-grained data access control in cloud computing**

To keep sensitive user data confidential against untrusted servers, existing solutions usually apply cryptographic methods by disclosing data decryption keys only to authorized users. However, in doing so, these solutions inevitably introduce a heavy computation overhead on the data owner for key distribution and data management when fine grained data access control is desired, and thus do not scale well. The problem of simultaneously achieving fine-grainedness scalability and data confidentiality of access control actually still remains unresolved. This paper proposed some services for data safekeeping and access control when users outsource sensitive data for sharing on cloud servers. This paper addresses this challenging open issue by, on one hand, defining and enforcing access policies based on data attributes, and, on the other hand, allowing the data owner to delegate most of the computation tasks involved in fine grained data access control to unfrosted cloud servers without disclosing the underlying data contents. Our proposed scheme enables the data owner to delegate tasks of data file re-encryption and user secret key update to cloud servers without disclosing data contents or user access privilege information. We achieve this goal by exploiting and uniquely combining techniques of attribute-based encryption (ABE), proxy reencryption, and lazy re-encryption. Our proposed scheme also has salient properties of user access privilege confidentiality and user secret key accountability and achieves fine - graininess, scalability and data confidentiality for data access control in cloud computing. Extensive analysis shows that our proposed scheme is highly efficient and provably secures under existing security models.

Fine-grained data access control in cloud computing. One challenge in this context is to achieve finegrainedness, data confidentiality, and scalability simultaneously, which is not provided by current work. In this paper we propose a scheme to achieve this goal by exploiting KPABE and uniquely combining it with techniques of proxy re-encryption and lazy re-encryption. Moreover, our proposed scheme can enable the data owner to delegate most of computation overhead to powerful cloud servers. Confidentiality of user access privilege and user secret key accountability can be achieved. Formal security proofs show that our proposed scheme is secure under standard cryptographic models.

**Effective Data Access Control for Multi-Authority Cloud Storage with Intrusion Detection**

Business Record (BR) is an emerging centric model of information exchange, which is often outsourced to be stored at a third party, such as cloud providers. However, there have been wide privacy concerns as business information could be exposed to those third party servers and to unauthorized parties. To assure the business data control over access to their own BRs, it is a promising method to encrypt the BRs before outsourcing. Yet, issues such as risks of privacy exposure, scalability in key management, flexible access and efficient user revocation, have remained the most important challenges toward achieving fine-grained, cryptographically enforced data access control. To achieve fine-grained and scalable data access control for BRs, we leverage attribute based encryption (ABE) techniques to encrypt all business file. We focus on the multiple data owner scenario, and divide the users in the BR system into multiple security domains that greatly reduces the key management complexity for owners and users. A high degree of data privacy is guaranteed simultaneously by exploiting multiauthority ABE.

The revocable multi-authority system has been implemented based on CP-ABE as the security criteria in data access control of the cloud storage. The storage system architectural environment resides like multi-stage or multiauthority based data access on business information. To reduce any security block whole, the system ensure the authentication by using attribute encryption rather than traditional data access as anonymously. The difficulties of revocation are reduced while applying attribute-based encryption by enabling multilevel independent authority for every level of data owner. Where data owner is the primary controller of the data and allows for requesting user to provide any attribute key for ensure reliable access. Middle level component reside for every access in Cloud data access. Rich level user interface is available to gather all information on authentication. Key attributes of a level has been verified and pool of data access from the cloud server is provided. Central data storage being as cloud data center and provides data access on request based it verification defined by data owner. Controller model has the component to take effective process on incoming requests. The multi-authority CP-ABE is a promising technique, which can be applied in any remote storage systems.

**3. ANALYSIS**

**Introduction**

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering), [information systems](http://en.wikipedia.org/wiki/Information_systems) and [software engineering](http://en.wikipedia.org/wiki/Software_engineering), is the process of creating or altering systems, and the models and [methodologies](http://en.wikipedia.org/wiki/Methodologies) that people use to develop these systems. In software engineering the SDLC concept underpins many kinds of [software development methodologies](http://en.wikipedia.org/wiki/Software_development_methodologies). These methodologies form the framework for planning and controlling the creation of an information system the [software development process](http://en.wikipedia.org/wiki/Software_development_process).

**Existing System**

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.

**Disadvantages of Existing System:**

1. 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.
2. 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.

**Proposed System**

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 owner’s effective user privilege management capability. 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 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.

**Advantages of Proposed system:**

1. Greatly reduce the computational overhead on client side mobile devices.
2. The lazy re-encryption and description field of attributes to reduce the revocation overhead when dealing with the user revocation problem.

**3.1. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

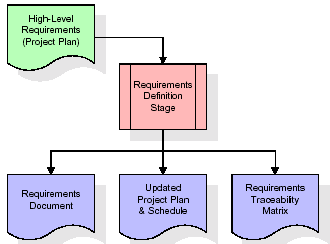
SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

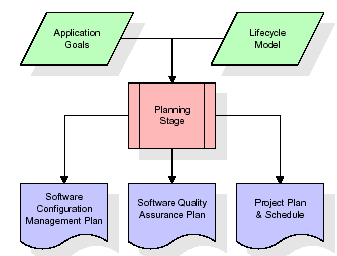
In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term *requirements traceability*.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

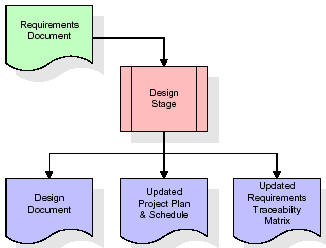
The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

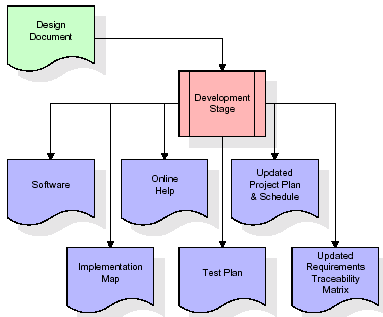
**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

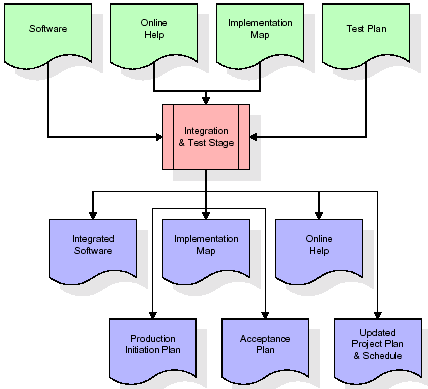
The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

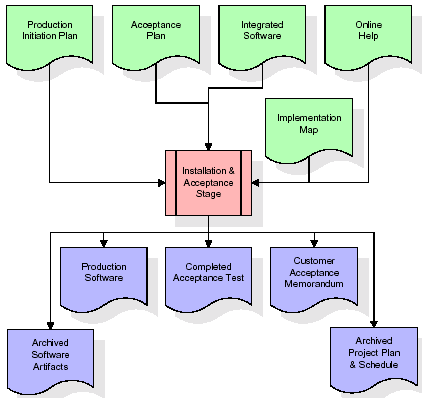


The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.2. Software Requirement Specification**

**3.2.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Nonfunctional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms *what* must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .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:
* **ECONOMIC 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.

* **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. 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.

* **TECHNICAL FEASIBILITY**

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.

**3.2.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly Java Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through Java capabilities.

**Software Interfaces**

The required software is JAVA1.6.

**Operating Environment**

Windows XP, Linux.

**HARDWARE REQUIREMENTS:**

# Processor - Pentium –IV

* Speed - 1.1 Ghz
* RAM - 256 MB(min)
* Hard Disk - 20 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**SOFTWARE REQUIREMENTS:**

* Operating System : Windows XP
* Programming Language : Java

**4. DESIGN**

**UML diagrams**

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* + **User Model View**
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.
  + **Structural Model view**
    1. In this model the data and functionality are arrived from inside the system.
    2. This model view models the static structures.
* **Behavioral Model View**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* **Implementation Model View**

In this the structural and behavioral as parts of the system are represented as they are to be built.

* **Environmental Model View**

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

**4.1 Class diagram:-**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. A class with three sections, in the diagram, classes is represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake

**Class diagram:**



**4.2 Use case diagram:-**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

**4.2.1 Use case diagram:**



**4.3. Sequence Diagram:**

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.

**4.3.1 Sequence diagram:**

****

**4.4 Collaboration diagram**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

**4.5.1 Collaboration diagram:**



**4.6 Component Diagram**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

**4.6.1 Component diagram:**



**4.7 Deployment Diagram**

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

**4.7.1 Deployment diagram:**



**4.8 Activity diagram:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It 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.

**4.8.1 Activity diagram:**

Register

Login

Enc-Dec-Server

Cloud Server

TA

Encrypt & Decrypt the files

Generate & send keys

Upload file

Download file

Overhead Graph

Logout

**4.9 Data Flow Diagram:**

[Data flow diagrams](http://www.edrawsoft.com/Data-Flow-Diagrams.php) illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

**Data Flow Diagram:**

7. Login the user

4. Register the users

2. Starts the servers

User

3. Servers started

6. Registration completed

1. Database started

Database

8. Upload the file

10. Download the file

12. Run overhead graph 13. Generated the graph

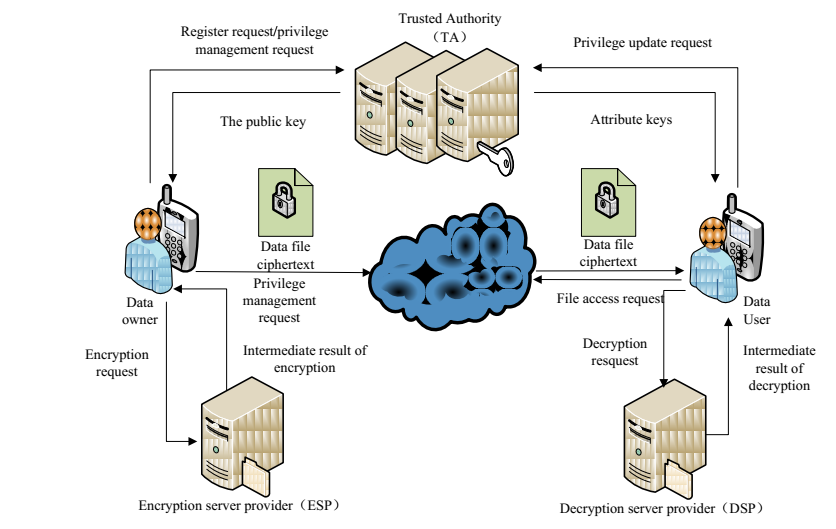
14. Exit the user 11. File is downloaded

9. File is uploaded

15. Display status on the servers

Cloud Server

**5. IMPLEMENTATION**



**Modules**

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)

**Module Description:**

**Data Owner (DO):**

DO uploads data to the mobile cloud and share it with friends. DO determines the access control policies.

**Data User (DU):**

DU retrieves data from the mobile cloud.

**Trust Authority (TA):**

TA is responsible for generating and distributing attribute keys.

**Encryption Service Provider (ESP):**

ESP provides data encryption operations for DO.

**Decryption Service Provider (DSP):**

DSP provides data decryption operations for DU.

**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 have stored in the cloud.

**5.1. Introduction of technologies used**

**About Java**:

Initially the language was called as “oak” but it was renamed as “java” in 1995.The primary motivation of this language was the need for a platform-independent (i.e. architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

* Java is a programmer’s language
* Java is cohesive and consistent
* Except for those constraint imposed by the Internet environment. Java gives the programmer, full control

Finally Java is to Internet Programming where c was to System Programming.

**Importance of Java to the Internet**

Java has had a profound effect on the Internet. This is because; java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the server and the personal computer. They are passive information and Dynamic active programs. in the areas of Security and probability. But Java addresses these concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

**Applications and applets**

An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++ .Java’s ability to create Applets makes it important. An Applet I san application, designed to be transmitted over the Internet and executed by a Java-compatible web browser. An applet I actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can be react to the user input and dynamically change.

**Java Architecture**

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

**Compilation of code**

When you compile the code, the Java compiler creates machine code (called byte code)for a hypothetical machine called Java Virtual Machine(JVM). The JVM is supposed t executed the byte code. The JVM is created for the overcoming the issue of probability. The code is written and compiled for one machine and interpreted on all machines .This machine is called Java Virtual Machine.

**Compiling and interpreting java source code.**

****

During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be an Intel Pentium windows 95 or sun SPARCstation running Solaris or Apple Macintosh running system and all could receive code from any computer through internet and run the Applets.

**Simple**:

Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ Programmer, learning Java will oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

### Object oriented

Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank state. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

### Robust

The multi-platform environment of the web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java? Java is strictly typed language; it checks your code at compile time and runtime.

Java virtually eliminates the problems of memory management and deal location, which is completely automatic. In a well-written Java program, all run-time errors can and should be managed by your program.

**AWT and Swings:**

**AWT:**

**Graphical User Interface:**

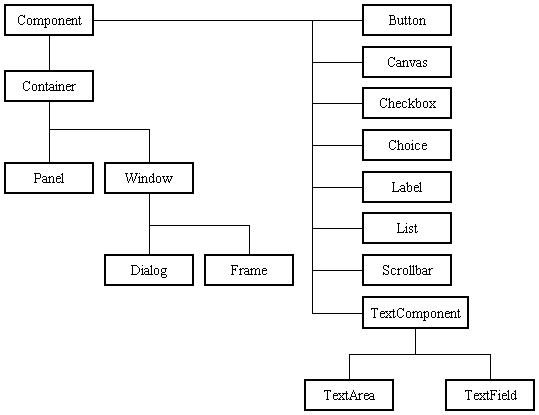
The user interface is that part of a program that interacts with the user of the program. GUI is a type of [user interface](http://en.wikipedia.org/wiki/User_interface) that allows [users](http://en.wikipedia.org/wiki/User_(computing)) to [interact](http://en.wikipedia.org/wiki/Human-computer_interaction) with electronic devices with images rather than text commands. A class library is provided by the Java programming language which is known as Abstract Window Toolkit (AWT) for writing graphical programs. The Abstract Window Toolkit (AWT) contains several graphical widgets which can be added and positioned to the display area with a layout manager.

As the Java programming language, the AWT is not platform-independent. AWT uses system peers object for constructing graphical widgets. A common set of tools is provided by the AWT for graphical user interface design. The implementation of the user interface elements provided by the AWT is done using every platform's native GUI toolkit. One of the AWT's significance is that the look and feel of each platform can be preserved.

**Components:**

A graphical user interface is built of graphical elements called components. A *component* is an object having a graphical representation that can be displayed on the screen and that can interact with the user. Components allow the user to interact with the program and provide the input to the program. In the AWT, all user interface components are instances of class Component or one of its subtypes. Typical components include such items as buttons, scrollbars, and text fields.

**Types of Components:**

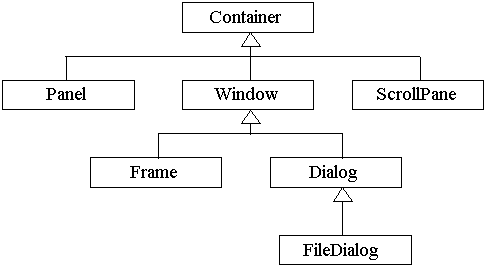
****

Before proceeding ahead, first we need to know what containers are. After learning containers we learn all components in detail.

**Containers:**

Components do not stand alone, but rather are found within containers. In order to make components visible, we need to add all components to the container. Containers contain and control the layout of components. In the AWT, all containers are instances of class Container or one of its subtypes. Components must fit completely within the container that contains them. For adding components to the container we will use add() method.

**Types of containers:**

****

**Basic GUI Logic:**

The GUI application or applet is created in three steps. These are:

* Add components to Container objects to make your GUI.
* Then you need to setup event handlers for the user interaction with GUI.
* Explicitly display the GUI for application.

A new thread is started by the interpreter for user interaction when an AWT GUI is displayed. When any event is received by this new thread such as click of a mouse, pressing of key etc then one of the event handlers is called by the new thread set up for GUI. One important point to note here is that the event handler code is executed within the thread.

**Creating a Frame:**

**Method1:**

In the first method we will be creating frame by extending Frame class which is defined in java.awt package. Following program demonstrate the creation of a frame.

import java.awt.\*;

public class FrameDemo1 extends Frame

{

FrameDemo1()

{

setTitle("Label Frame");

setVisible(true);

setSize(500,500);

}

public static void main(String[] args)

{

new FrameDemo1 ();

}

}

In the above program we are using three methods:

setTitle: For setting the title of the frame we will use this method. It takes String as an argument which will be the title name.

SetVisible: For making our frame visible we will use this method. This method takes Boolean value as an argument. If we are passing true then window will be visible otherwise window will not be visible.

SetSize: For setting the size of the window we will use this method. The first argument is width of the frame and second argument is height of the frame.

**Method 2:**

In this method we will be creating the Frame class instance for creating frame window. Following program demonstrate Method2.

import java.awt.\*;

public class FrameDemo2

{

public static void main(String[] args)

{

Frame f = new Frame();

f.setTitle("My first frame");

f.setVisible(true);

f.setSize(500,500);

}

}

**Types of Components:**

1. **Labels :**

This is the simplest component of Java Abstract Window Toolkit. This component is generally used to show the text or string in your application and label never perform any type of action.

Label l1 = new Label("One");

Label l2 = new Label("Two");

Label l3 = new Label("Three",Label.CENTER);

In the above three lines we have created three labels with the name “one, two, three”. In the third label we are passing two arguments. Second argument is the justification of the label. Now after creating components we will be adding it to the container.

add(l1);

add(l2);

add(l3);

We can set or change the text in a label by using the **setText( )** method. You can obtain the current label by calling **getText( )**. These methods are shown here:

void setText(String *str*)

String getText( )

1. **Buttons :**

This is the component of Java Abstract Window Toolkit and is used to trigger actions and other events required for your application. The syntax of defining the button is as follows :

Button l1 = new Button("One");

Button l2 = new Button("Two");

Button l3 = new Button("Three");

We can change the Button's label or get the label's text by using the Button.setLabel(String) and Button.getLabel() method.

1. **CheckBox:**

A *check box is a control that is used to turn an option on or off. It consists of a small box* that can either contain a check mark or not. There is a label associated with each check box that describes what option the box represents. You change the state of a check box by clicking on it. The syntax of the definition of Checkbox is as follows :

Checkbox Win98 = new Checkbox("Windows 98/XP", null, true);

Checkbox winNT = new Checkbox("Windows NT/2000");

Checkbox solaris = new Checkbox("Solaris");

Checkbox mac = new Checkbox("MacOS");

The first form creates a check box whose label is specified in first argument and whose group is specified in second argument*.* If this check box is not part of a group, then *cbGroup* must be **null**. (Check box groups are described in the next section.) The value truedetermines the initial state of the check box is checked. The second form creates a check box with only one parameter.

To retrieve the current state of a check box, call **getState( )**. To set its state, call **setState( )**. You can obtain the current label associated with a check box by calling **getLabel( )**. To set the label, call **setLabel( )**. These methods are as follows:

boolean getState( )

void setState(boolean *on*)

String getLabel( )

void setLabel(String *str*)

Here, if *on* is **true**, the box is checked. If it is **false**, the box is cleared. The string passed in *str* becomes the new label associated with the invoking check box.

1. **Radio Button:**

This is the special case of the Checkbox component of Java AWT package. This is used as a group of checkboxes which group name is same. Only one Checkbox from a Checkbox Group can be selected at a time. Syntax for creating radio buttons is as follows:

CheckboxGroup cbg = new CheckboxGroup();

Checkbox Win98 = new Checkbox("Windows 98/XP", cbg , true);

Checkbox winNT = new Checkbox("Windows NT/2000",cbg, false);

Checkbox solaris = new Checkbox("Solaris",cbg, false);

Checkbox mac = new Checkbox("MacOS",cbg, false);

For Radio Button we will be using CheckBox class. The only difference in Checkboxes and radio button is in Check boxes we will specify null for checkboxgroup but whereas in radio button we will be specifiying the checkboxgroup object in the second parameter.

1. **Choice:**

The Choice class is used to create a pop-up list of items from which the user may choose. Thus, a Choice control is a form of menu. Syntax for creating choice is as follows:

Choice os = new Choice();

/\* adding items to choice \*/

os.add("Windows 98/XP");

os.add("Windows NT/2000");

os.add("Solaris");

os.add("MacOS");

We will be creating choice with the help of Choice class. Pop up list will be creating with the creation of object, but it will not have any items. For adding items we will be using add() method defined in Choice class.

To determine which item is currently selected, you may call either **getSelectedItem( )** or **getSelectedIndex( )**. These methods are shown here:

String getSelectedItem( )

int getSelectedIndex( )

The **getSelectedItem( )** method returns a string containing the name of the item. **getSelectedIndex( )** returns the index of the item. The first item is at index 0. By default, the first item added to the list is selected.

1. **List:**

List class is also same as choice but the only difference in list and choice is, in choice user can select only one item whereas in List user can select more than one item. Syntax for creating list is as follows:

List os = new List(4, true);

First argument in the List constructor specifies the number of items allowed in the list. Second argument specifies whether multiple selections are allowed or not.

/\* Adding items to the list \*/

os.add("Windows 98/XP");

os.add("Windows NT/2000");

os.add("Solaris");

os.add("MacOS");

In list we can retrieve the items which are selected by the users. In multiple selection user will be selecting multiple values for retrieving all the values we have a method called getSelectedValues() whose return type is string array. For retrieving single value again we can use the method defined in Choice i.e. getSelectedItem().

1. **TextField:**

Text fields allow the user to enter strings and to edit the text using the arrow keys, cut and paste keys. TextField is a subclass of TextComponent. Syntax for creating list is as follows:

TextField tf1 = new TextField(25);

TextField tf2 = new TextField();

In the first text field we are specifying the size of the text field and the second text field is created with the default value. **TextField** (and its superclass **TextComponent**) provides several methods that allow you to utilize a text field. To obtain the string currently contained in the text field, call **getText( )**. To set the text, call **setText( )**. These methods are as follows:

String getText( )

void setText(String *str*)

We can control whether the contents of a text field may be modified by the user by calling **setEditable( )**. You can determine editability by calling **isEditable( )**. These methods are shown here:

boolean isEditable( )

void setEditable(boolean *canEdit*)

**isEditable( )** returns **true** if the text may be changed and **false** if not. In **setEditable( )**, if *canEdit* is **true**, the text may be changed. If it is **false**, the text cannot be altered.

There may be times when we will want the user to enter text that is not displayed, such as a password. We can disable the echoing of the characters as they are typed by calling **setEchoChar( )**.

1. **TextArea:**

TextArea is a multiple line editor. Syntax for creating list is as follows:

TextArea area = new TextArea(20,30);

Above code will create one text area with 20 rows and 30 columns. **TextArea** is a subclass of **TextComponent**. Therefore, it supports the **getText( )**, **setText( )**, **getSelectedText( )**, **select( )**, **isEditable( )**, and **setEditable( )** methods described in the preceding section.

**TextArea** adds the following methods:

void append(String *str*)

void insert(String *str*, int *index*)

void replaceRange(String *str*, int *startIndex*, int *endIndex*)

The **append( )** method appends the string specified by *str* to the end of the current text. **insert( )** inserts the string passed in *str* at the specified index. To replace text, call **replaceRange( )**. It replaces the characters from *startIndex* to *endIndex*–1, with the replacement text passed in *str.*

**Layout Managers:**

A layout manager automatically arranges controls within a window by using some type of algorithm. Each **Container** object has a layout manager associated with it. A layout manager is an instance of any class that implements the **LayoutManager** interface. The layout manager is set by the **setLayout( )** method. If no call to **setLayout( )** is made, then the default layout manager is used. Whenever a container is resized (or sized for the first time), the layout manager is used to position each of the components within it. The **setLayout( )** method has the following general form:

void setLayout(LayoutManager *layoutObj*)

Here, *layoutObj* is a reference to the desired layout manager. If you wish to disable the layout manager and position components manually, pass **null** for *layoutObj.* If we do this, you will need to determine the shape and position of each component manually, using the setBounds( ) method defined by Component.

Void setBounds(int x , int y , int width, int length)

In which first two arguments are the x and y axis. Third argument is width and fourth argument is height of the component.

Java has several predefined **LayoutManager** classes, several of which are described next. You can use the layout manager that best fits your application.

**FlowLayout:**

**FlowLayout** is the default layout manager. This is the layout manager that the preceding examples have used. **FlowLayout** implements a simple layout style, which is similar to how words flow in a text editor. Components are laid out from the upper-left corner, left to right and top to bottom. When no more components fit on a line, the next one appears on the next line. A small space is left between each component, above and below, as well as left and right. Here are the constructors for **FlowLayout**:

FlowLayout( )

FlowLayout(int *how*)

FlowLayout(int *how*, int *horz*, int *vert*)

The first form creates the default layout, which centers components and leaves five pixels of space between each component. The second form lets you specify how each line is aligned. Valid values for *how* are as follows:

FlowLayout.LEFT

FlowLayout.CENTER

FlowLayout.RIGHT

These values specify left, center, and right alignment, respectively. The third form allows you to specify the horizontal and vertical space left between components in *horz* and *vert,* respectively.

**BorderLayout:**

THE JAVA LIBRARYThe **BorderLayout** class implements a common layout style for top-level windows. It has four narrow, fixed-width components at the edges and one large area in the center. The four sides are referred to as north, south, east, and west. The middle area is called the center. Here are the constructors defined by **BorderLayout**:

BorderLayout( )

BorderLayout(int *horz*, int *vert*)

The first form creates a default border layout. The second allows you to specify the horizontal and vertical space left between components in *horz* and *vert,* respectively. **BorderLayout** defines the following constants that specify the regions:

BorderLayout.CENTER BorderLayout.SOUTH

BorderLayout.EAST BorderLayout.WEST

BorderLayout.NORTH

When adding components, you will use these constants with the following form of **add( )**, which is defined by **Container**:

void add(Component *compObj,* Object *region*);

Here, *compObj* is the component to be added, and *region* specifies where the component will be added.

**GridLayout:**

**GridLayout** lays out components in a two-dimensional grid. When you instantiate a **GridLayout**, you define the number of rows and columns. The constructors supported by **GridLayout** are shown here:

GridLayout( )

GridLayout(int *numRows*, int *numColumns* )

GridLayout(int *numRows*, int *numColumns*, int *horz*, int *vert*)

The first form creates a single-column grid layout. The second form creates a grid layout with the specified number of rows and columns. The third form allows you to specify the horizontal and vertical space left between components in *horz* and *vert*, respectively. Either *numRows* or *numColumns* can be zero. Specifying *numRows* as zero allows for unlimited-length columns. Specifying *numColumns* as zero allows for unlimited-length rows.

**Swings:**

**About Swings:**

Swing is important to develop Java programs with a graphical user interface (GUI). There are many components which are used for the building of GUI in Swing. The Swing Toolkit consists of many components for the building of GUI. These components are also helpful in providing interactivity to Java applications. Following are components which are included in Swing toolkit:

* list controls
* buttons
* labels
* tree controls
* table controls

All AWT flexible components can be handled by the Java Swing. Swing toolkit contains far more components than the simple component toolkit. It is unique to any other toolkit in the way that it supports integrated internationalization, a highly customizable text package, rich undo support etc. Not only this you can also create your own look and feel using Swing other than the ones that are supported by it. The customized look and feel can be created using Synth which is specially designed. Not to forget that Swing also contains the basic user interface such as customizable painting, event handling, drag and drop etc.

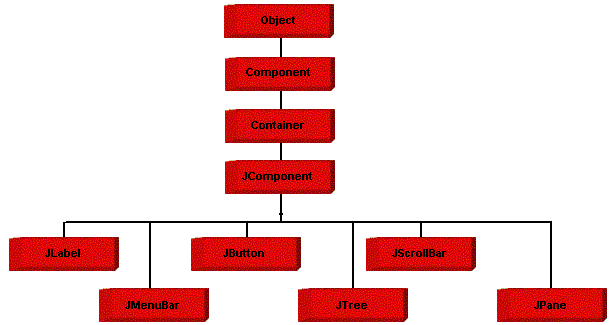
The Java Foundation Classes (JFC) which supports many more features important to a GUI program comprises of Swing as well. The features which are supported by Java Foundation Classes (JFC) are the ability to create a program that can work in different languages, the ability to add rich graphics functionality etc.

There are several components contained in Swing toolkit such as check boxes, buttons, tables, text etc. Some very simple components also provide sophisticated functionality. For instance, text fields provide formatted text input or password field behavior. Furthermore, the file browsers and dialogs can be used according to one's need and can even be customized.

**Difference between Swings and AWT:**

|  |  |
| --- | --- |
| **Swings** | **AWT** |
| Swings are the light weight components. | AWTs are the heavy weight components. |
| Swings are developed by using pure java language. | AWTs are developed by using C and C++. |
| We can have different look and feel in swings. | This feature is not available in awt. |
| Swing has many advanced features like JTabel, JTabbedPane and JTree | This is not available in AWT. |

**Java Swing Class Hierarchy:**



**Swing Components:**

All the components which are supported in AWT same components are also supported in Swings with a slight change in their class name.

|  |  |
| --- | --- |
| **AWT Components** | **Swing Components** |
| Label | JLabel |
| TextField | JTextField |
| TextArea | JTextArea |
| Choice | JComboBox |
| Checkbox | JCheckBox |
| List | JList |
| Button | JButton |
| - | JRadioButton |
| - | JPasswordField |
| - | JTable |
| - | JTree |
| - | JTabbedPane |
| MenuBar | JMenuBar |
| Menu | JMenu |
| MenuItem | JMenuItem |
| - | JFileChooser |
| - | JOptionPane |

We will discuss only those components which are not discussed in AWT chapter.

**JTabbedPane class:**

 The JTabbedPane container allows many panels to occupy the same area of the interface, and the user may select which to show by clicking on a tab.

**Constructor**

JTabbedPane tp = new JTabbedPane();

## Adding tabs to the JTabbedPane

Add tabs to a tabbed pane by calling addTab and passing it a String title and an instance of a class which should be called when we pressed a tab. That class should be a subclass of JPanel.

addTab(“String”,instance);

**Example program:**

import javax.swing.\*;

import java.awt.\*;

public class TabbedPaneDemo extends JFrame

{

TabbedPaneDemo()

{

setLayout(new FlowLayout(FlowLayout.LEFT));

setTitle("Tabbed Demo");

setVisible(true);

setSize(500,500);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JTabbedPane pane = new JTabbedPane();

pane.addTab("Countries",new Count());

pane.addTab("Cities",new Cit());

add(pane);

}

public static void main(String a[])

{

new TabbedPaneDemo();

}

}

class Count extends JPanel

{

Count()

{

JButton b1 = new JButton("India");

JButton b2 = new JButton("SriLanka");

JButton b3 = new JButton("Australia");

add(b1);

add(b2);

add(b3);

}

}

class Cit extends JPanel

{

Cit()

{

JCheckBox cb1 = new JCheckBox("Hyderabad");

JCheckBox cb2 = new JCheckBox("Banglore");

JCheckBox cb3 = new JCheckBox("Pune");

add(cb1);

add(cb2);

add(cb3);

}

}

**JMenuBar, JMenu, JMenuItem**

A top-level window can have a menu bar associated with it. A menu bar displays a list of top-level menu choices. Each choice is associated with a drop-down menu. This concept is implemented in Java by the following classes: JMenuBar, JMenu, and JMenuItem. In general, a menu bar contains one or more JMenu objects. Each JMenu object contains a list of JMenuItem objects. Each JMenuItem object represents something that can be selected by the user. To create a menu bar, first create an instance of JMenuBar. This class only defines the default constructor. Next, create instances of JMenu that will define the selections displayed on the bar. Following are the constructors for Menu:

JMenu( )

JMenu(String *optionName*)

Here, *optionName* specifies the name of the menu selection. The first form creates an empty menu. Individual menu items are of type MenuItem. It defines these constructors:

JMenuItem( )

JMenuItem(String *itemName*)

Here, *itemName* is the name shown in the menu.

**5.5 Sample Code**

**CloudServer.jsp**

package com;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Container;

import java.awt.Font;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.UIManager;

import javax.swing.JTextArea;

import javax.swing.JScrollPane;

import java.net.Socket;

import java.net.ServerSocket;

import java.net.InetAddress;

import javax.swing.JOptionPane;

public class CloudServer extends JFrame implements Runnable{

JLabel l1;

Font f1,f2;

JPanel p1,p2,p3;

Thread thread;

JTextArea area;

JScrollPane jsp;

ServerSocket server;

RequestHandler rh;

static double time;

public void start(){

try{

server = new ServerSocket(1111);

area.append("Cloud Service Provider Started\n\n");

while(true){

Socket socket = server.accept();

socket.setKeepAlive(true);

InetAddress address=socket.getInetAddress();

String ipadd=address.toString();

area.append("Connected Computers :"+ipadd.substring(1,ipadd.length())+"\n");

rh = new RequestHandler(socket,area);

rh.start();

}

}catch(Exception e){

e.printStackTrace();

}

}

public CloudServer(){

setTitle("Cloud Service Storage Provider");

getContentPane().setLayout(new BorderLayout());

f1 = new Font("Monospaced",Font.BOLD,22);

p1 = new JPanel();

l1 = new JLabel("<HTML><BODY><CENTER>Cloud Service Provider Application</CENTER></BODY></HTML>".toUpperCase());

l1.setFont(this.f1);

l1.setForeground(new Color(125,54,2));

p1.setBackground(Color.black);

p1.add(l1);

f2 = new Font("Courier New",Font.PLAIN,16);

p2 = new JPanel();

p2.setLayout(new BorderLayout());

area = new JTextArea();

area.setFont(f2);

area.setEditable(false);

jsp = new JScrollPane(area);

p2.add(jsp,BorderLayout.CENTER);

getContentPane().add(p1, BorderLayout.NORTH);

getContentPane().add(p2, BorderLayout.CENTER);

thread = new Thread(this);

thread.start();

}

public void run(){

try{

while(true){

l1.setForeground(Color.white);

thread.sleep(500);

l1.setForeground(Color.red);

thread.sleep(500);

}

}catch(Exception e){

e.printStackTrace();

}

}

public static void main(String a[])throws Exception{

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

CloudServer screen = new CloudServer();

screen.pack();

screen.setExtendedState(JFrame.MAXIMIZED\_BOTH);

screen.setVisible(true);

new ServerThread(screen);

}

}

**TrustedAuthority.java**

package com;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Container;

import java.awt.Font;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.UIManager;

import javax.swing.JTextArea;

import javax.swing.JScrollPane;

import java.net.Socket;

import java.net.ServerSocket;

import java.net.InetAddress;

import javax.swing.JButton;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import javax.swing.JOptionPane;

import java.io.File;

public class TrustedAuthority extends JFrame implements Runnable{

JLabel l1;

Font f1,f2;

JPanel p1,p2;

Thread thread;

JTextArea area;

JScrollPane jsp;

ServerSocket server;

RequestHandler rh;

public void start(){

try{

server = new ServerSocket(2222);

area.append("Trusted Authority Center Started\n\n");

while(true){

Socket socket = server.accept();

socket.setKeepAlive(true);

InetAddress address=socket.getInetAddress();

String ipadd=address.toString();

area.append("Connected Computers :"+ipadd.substring(1,ipadd.length())+"\n");

rh = new RequestHandler(socket,area);

rh.start();

}

}catch(Exception e){

e.printStackTrace();

}

}

public TrustedAuthority(){

setTitle("Trusted Authority Center");

getContentPane().setLayout(new BorderLayout());

f1 = new Font("Monospaced",Font.BOLD,22);

p1 = new JPanel();

l1 = new JLabel("<HTML><BODY><CENTER>Trusted Authority Key Generation Application</CENTER></BODY></HTML>".toUpperCase());

l1.setFont(this.f1);

l1.setForeground(new Color(125,54,2));

p1.setBackground(Color.black);

p1.add(l1);

f2 = new Font("Courier New",Font.PLAIN,16);

p2 = new JPanel();

p2.setLayout(new BorderLayout());

area = new JTextArea();

area.setFont(f2);

area.setEditable(false);

jsp = new JScrollPane(area);

p2.add(jsp,BorderLayout.CENTER);

getContentPane().add(p1, BorderLayout.NORTH);

getContentPane().add(p2, BorderLayout.CENTER);

thread = new Thread(this);

thread.start();

}

public void run(){

try{

while(true){

l1.setForeground(Color.white);

thread.sleep(500);

l1.setForeground(Color.red);

thread.sleep(500);

}

}catch(Exception e){

e.printStackTrace();

}

}

public static void main(String a[])throws Exception{

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

TrustedAuthority screen = new TrustedAuthority();

screen.pack();

screen.setExtendedState(JFrame.MAXIMIZED\_BOTH);

screen.setVisible(true);

new ServerThread(screen);

}

}

**6. TESTING**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

## Implementation

## The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

**TEST CASES:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | Upload file | Verify either file is uploaded or not | If file is not uploaded | We cannot share the data | file is uploaded | High | High |
| 02 | Download file | whether the file is downloaded or not | If it’s not downloaded | We cannot get the files | File is downloaded | High | High |
| 03 | Generate the graph | Whether the graph is generated or not | If it’s not generated | We cannot get the encryption files | Graph is generated | High | High |
| 04 | exit | Whether user is exit or not | If he not exit | He cannot complete the process | He has complete the process | High | High |

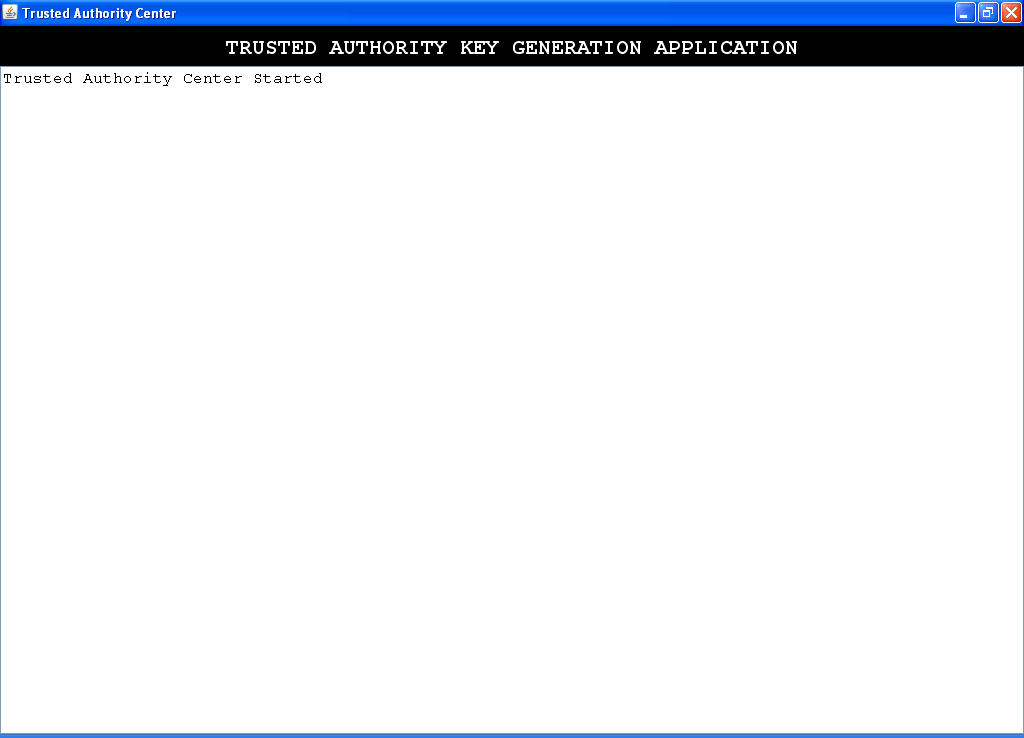
**7. SCREEN SHOTS**

Screen shots

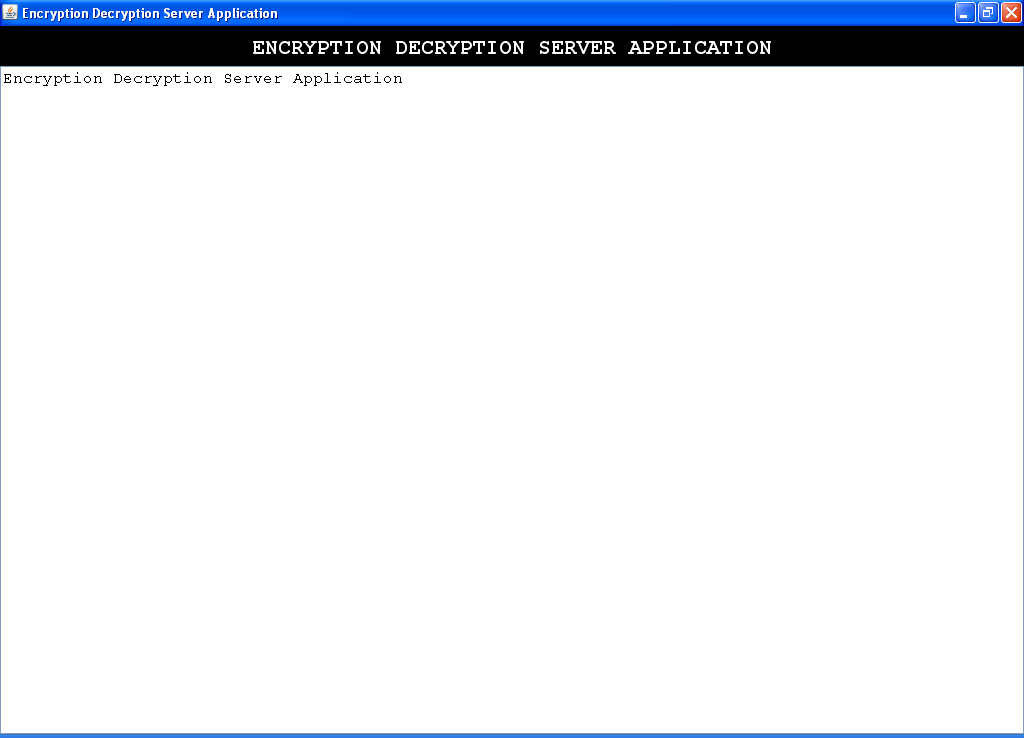
Double click on ‘run.bat’ file from LDSS/WEB-INF/CloudServer folder to start cloud server below screen



Double click on ‘run.bat’ file from LDSS/WEB-INF/TA folder to start Trusted Authority



Double click on ‘run.bat’ file from LDSS/WEB-INF/ Enc\_Dec\_Server folder to start encryption and decryption server



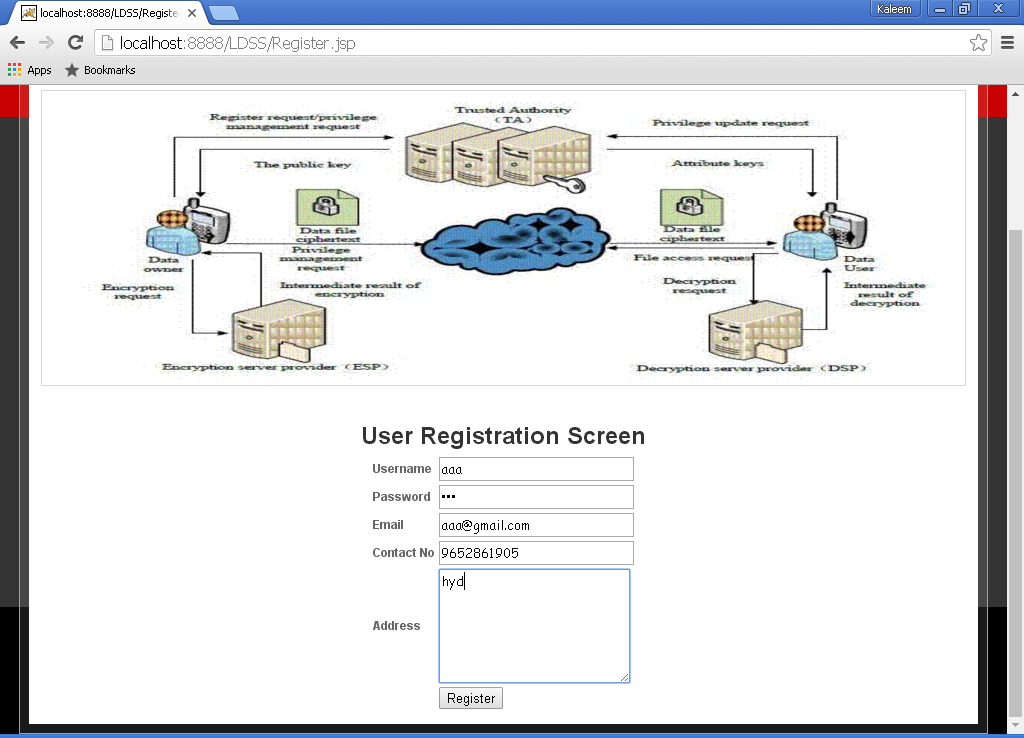
Now start tomcat server and open browser and enter URL as below

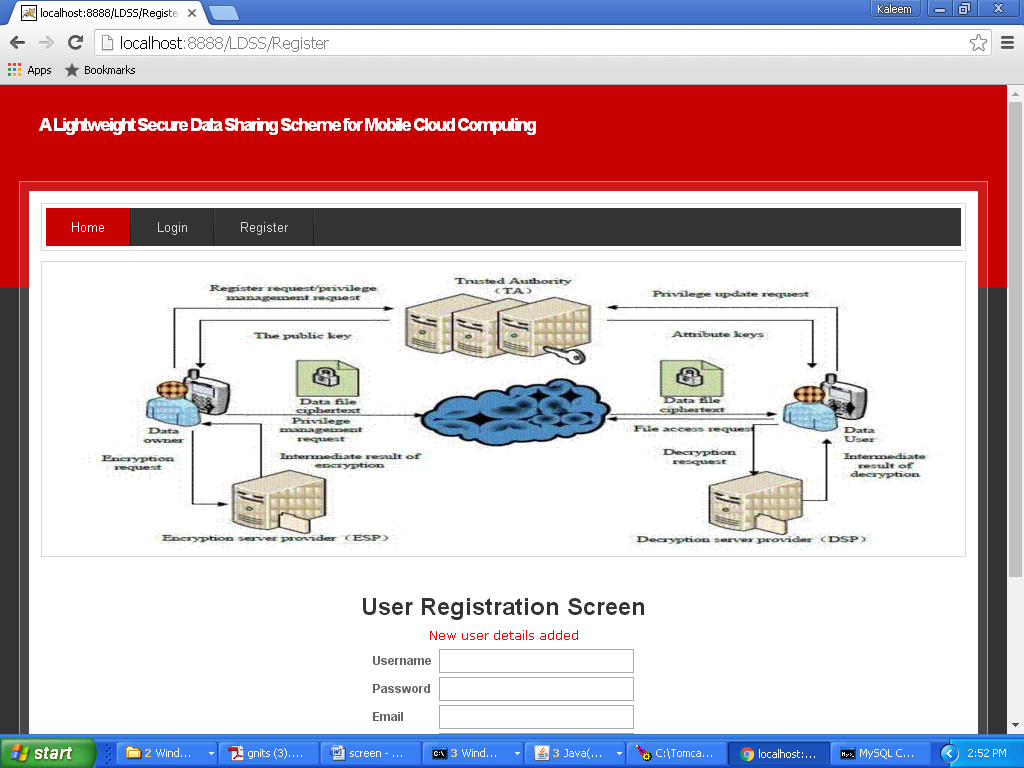
<http://localhost:tomcat_server_port/LDSS>

will get below screen

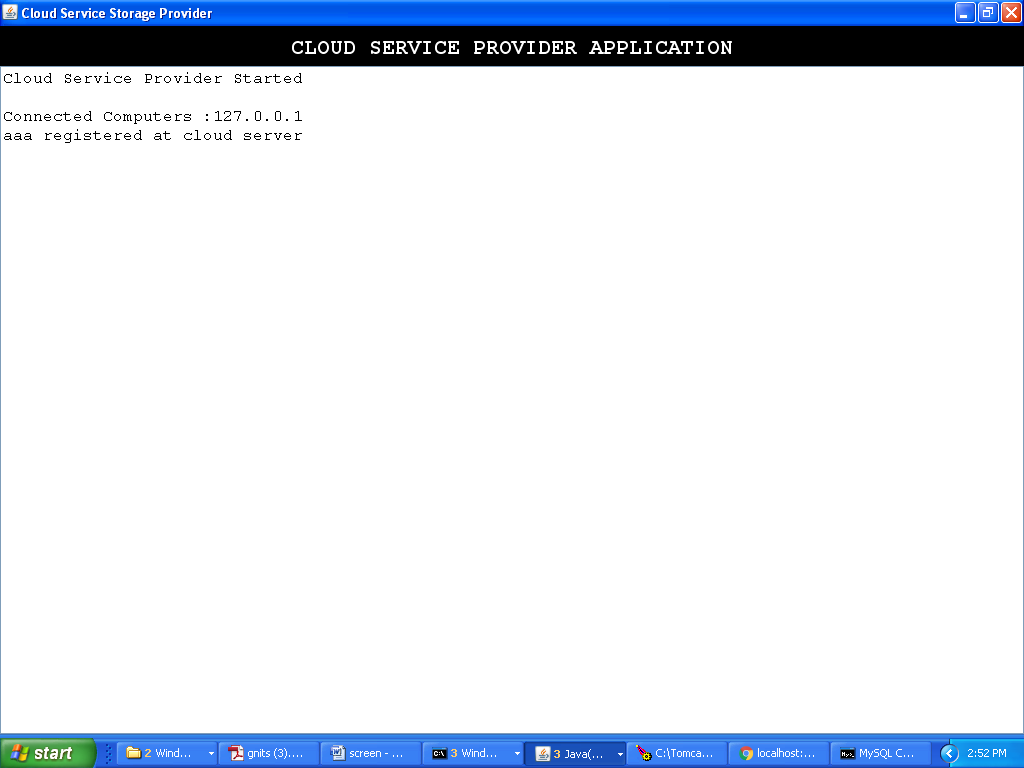


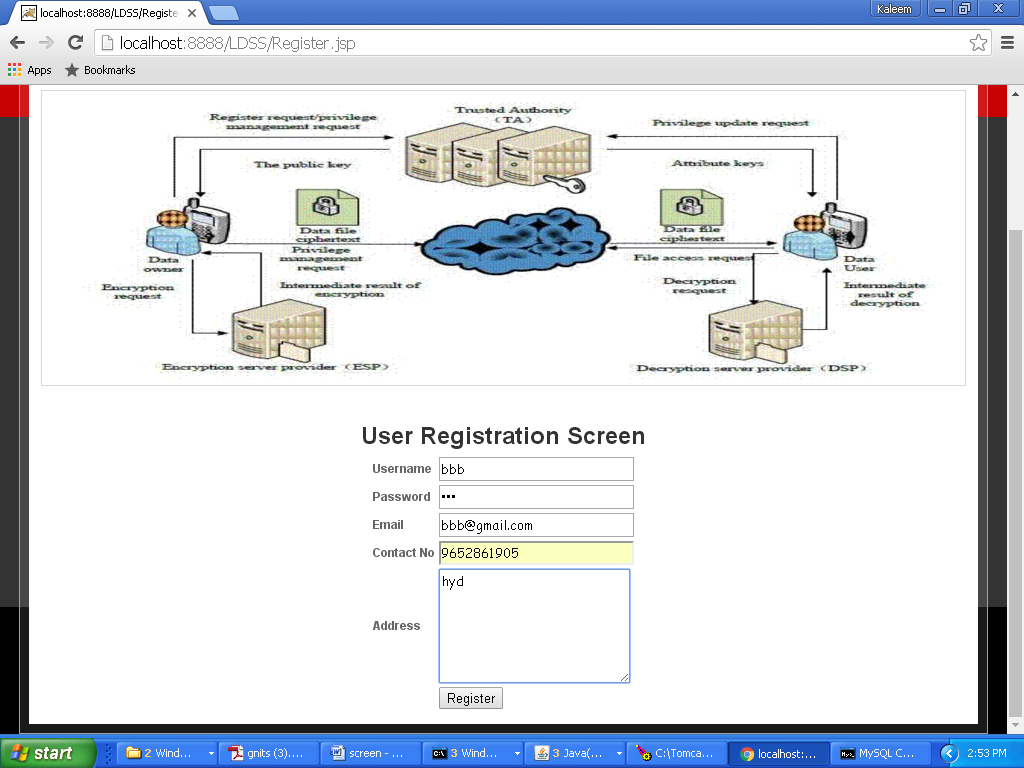
Now click on Register link and add some users

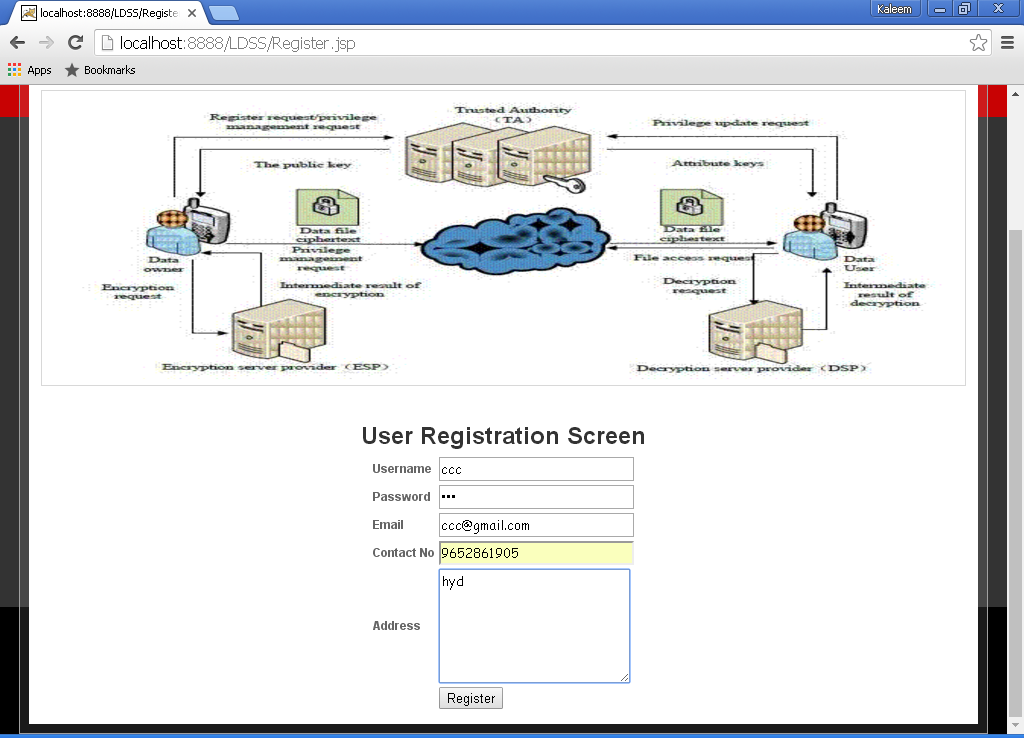




Will get user information at cloud server also







I added three users now login as one user and upload and share data



Will get below screen after successful login

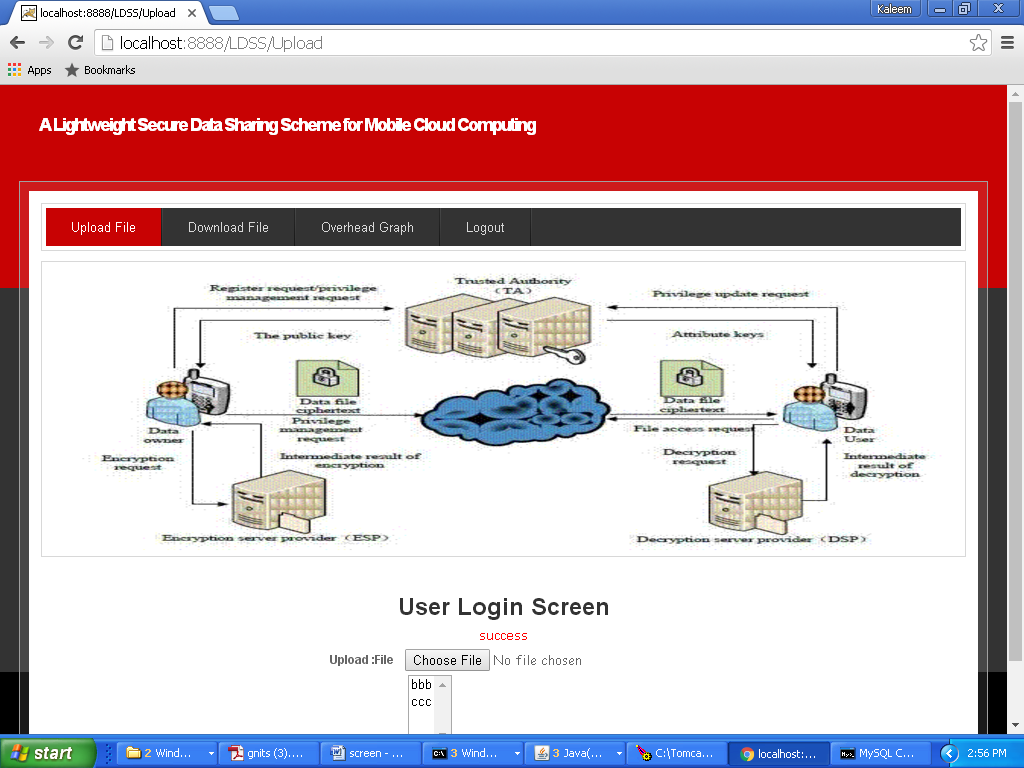


Click on ‘Upload File’ link to upload data

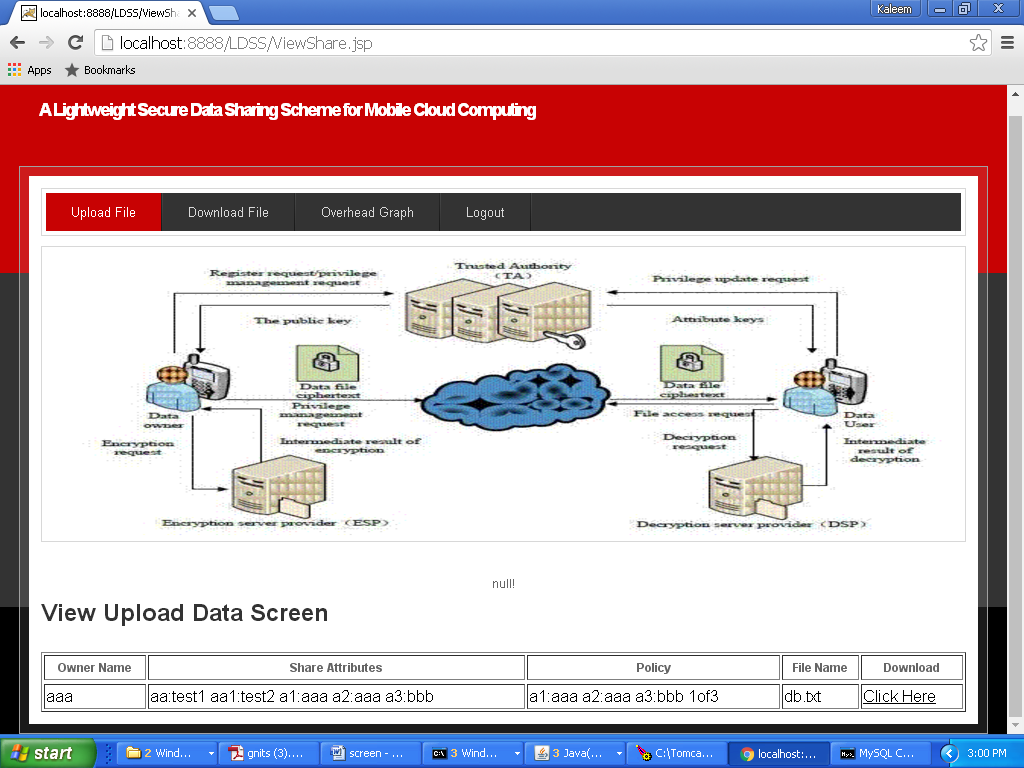




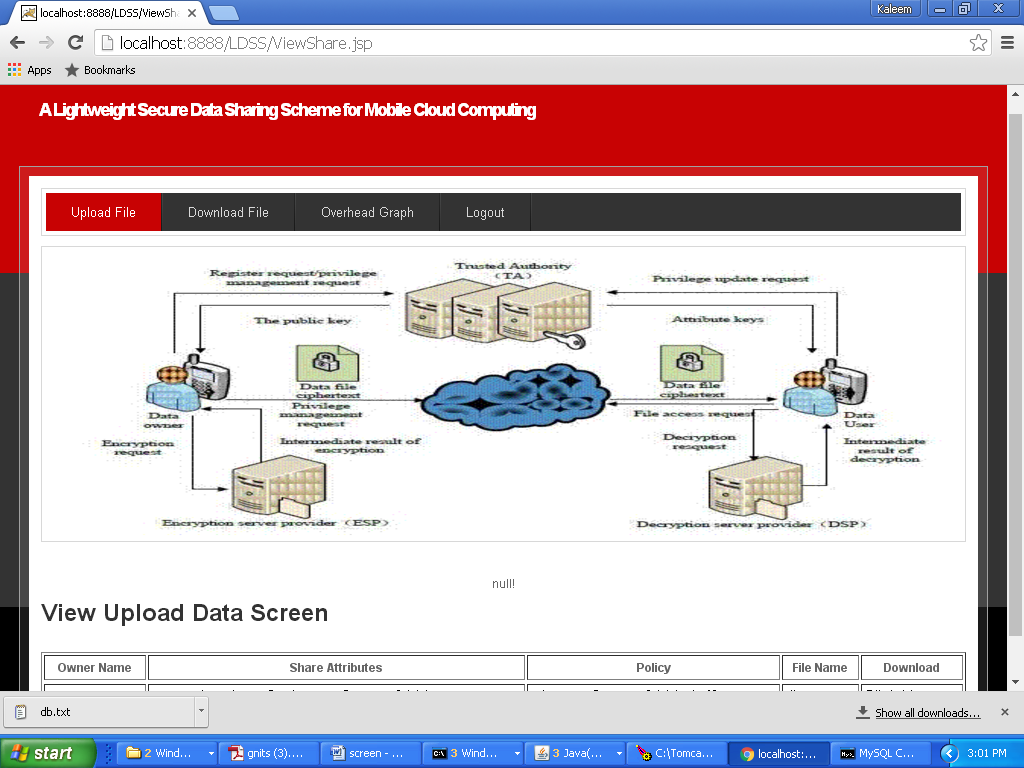
In above screen user aaa upload ‘db.txt’ file and sharing with user bbb



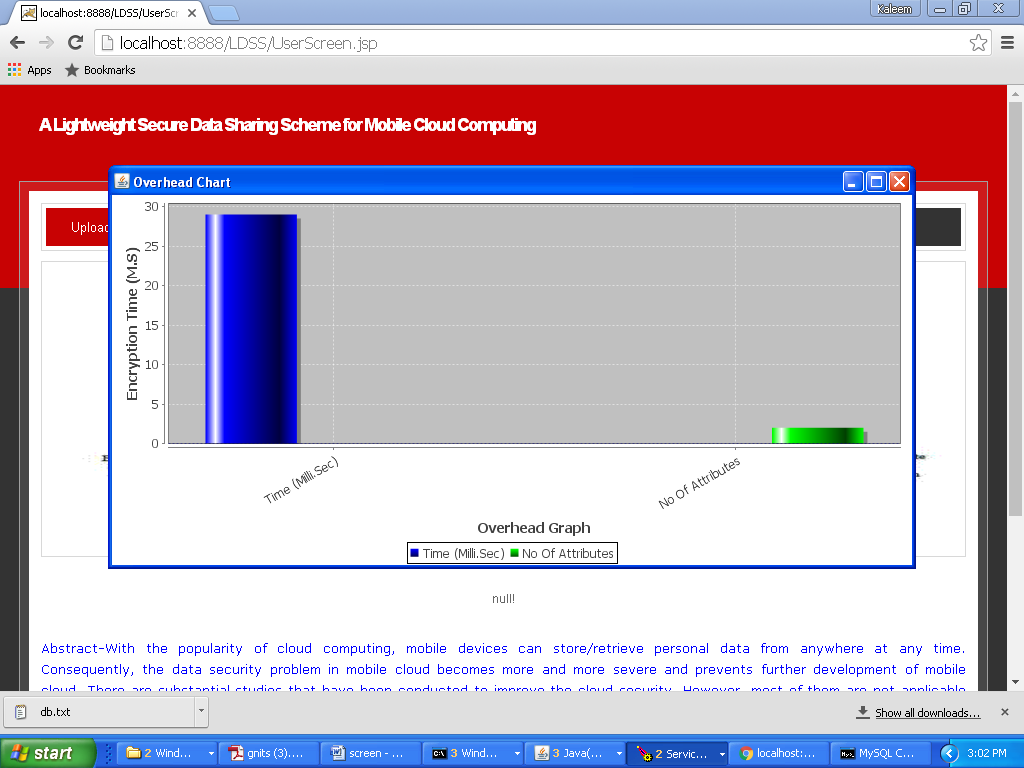
User aaa and user bbb can download the file but not user ccc. Now click on ‘Download File’ link to view share and to download file.



Now click on ‘Click Here’ to link to download file



In browser task bar we can see db.txt file downloaded. Now click on ‘Over head Graph link’ to generate graph

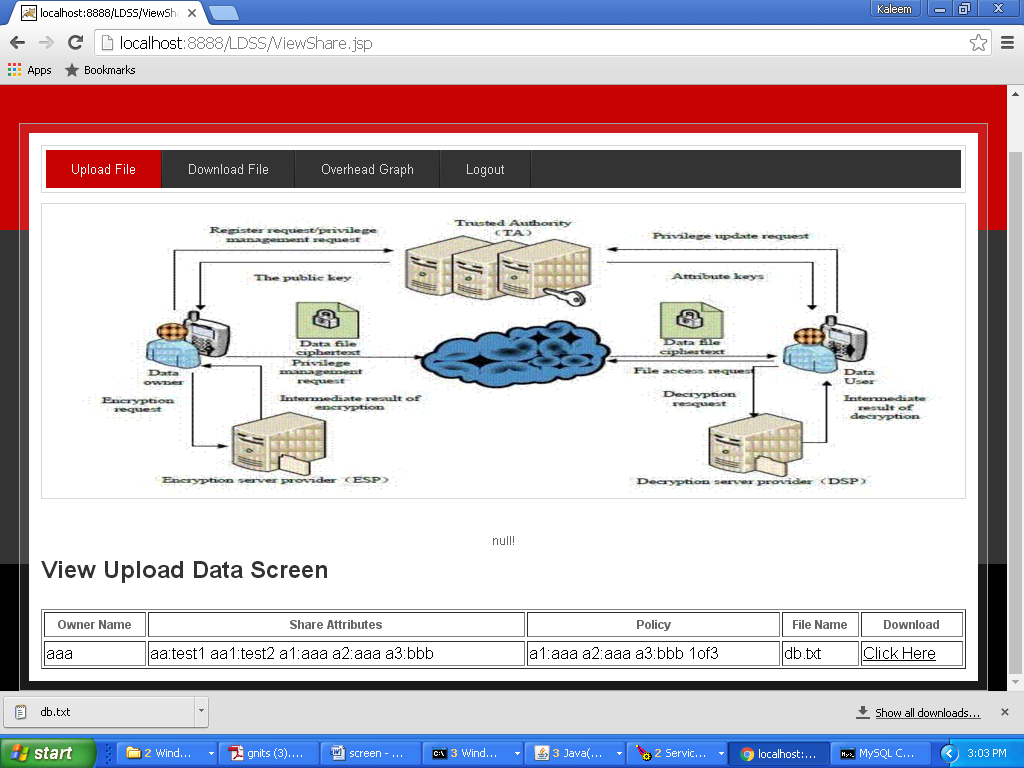


Now logout and login as ccc

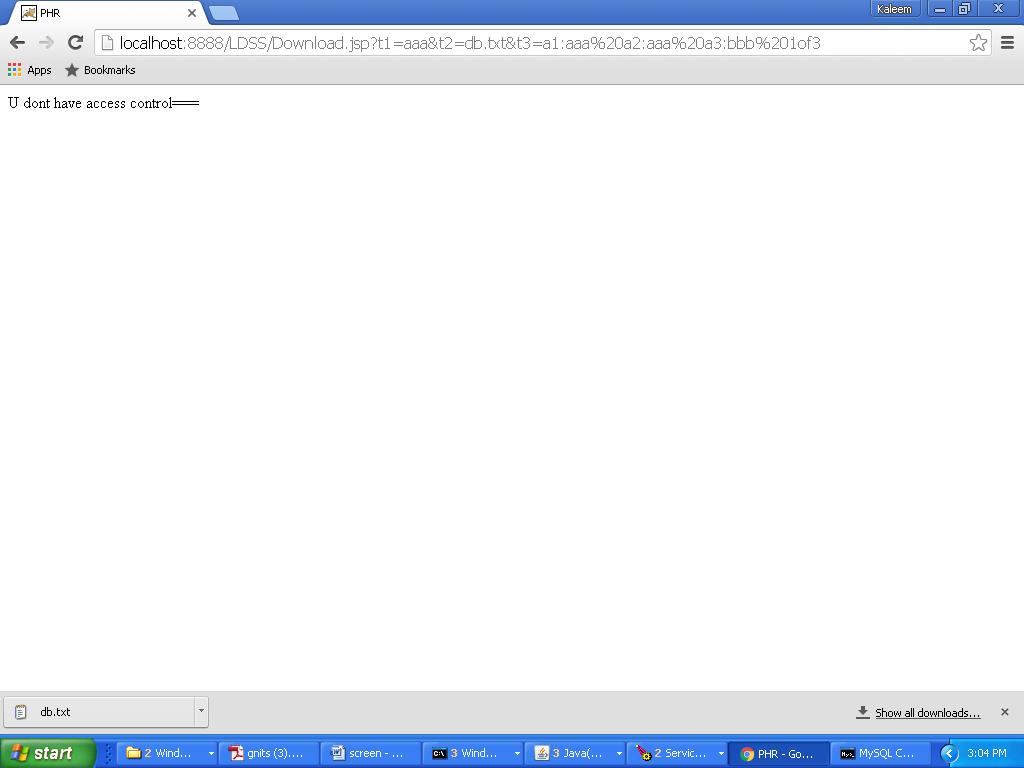




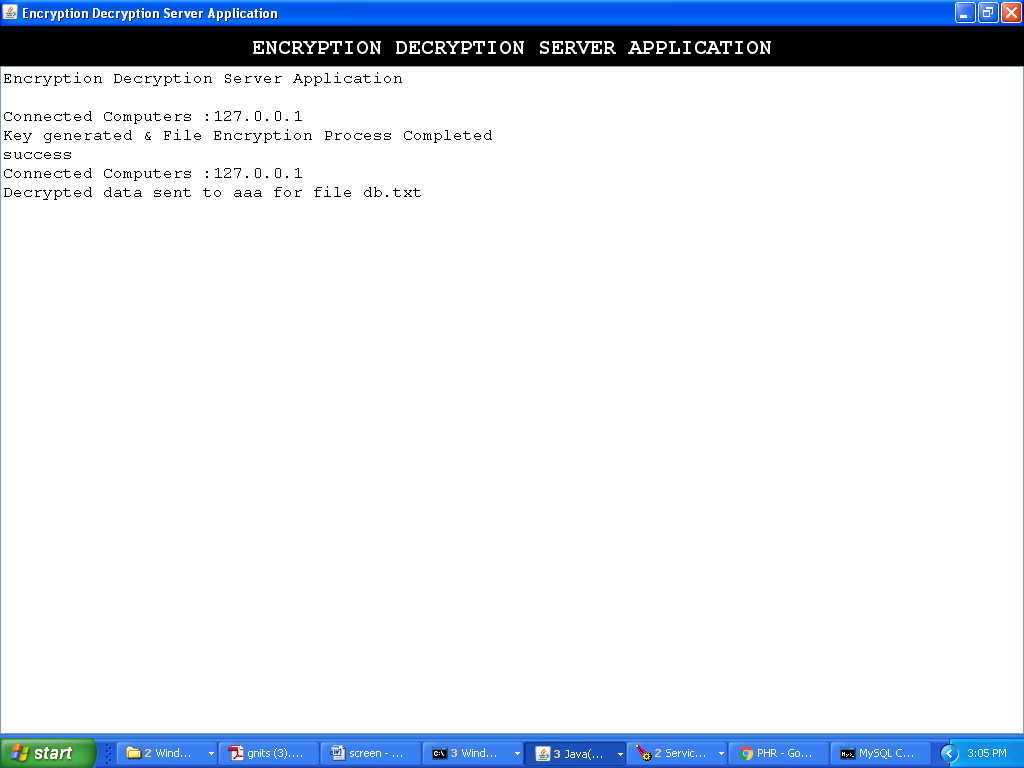
Now click on ‘Download File’ link to get below screen



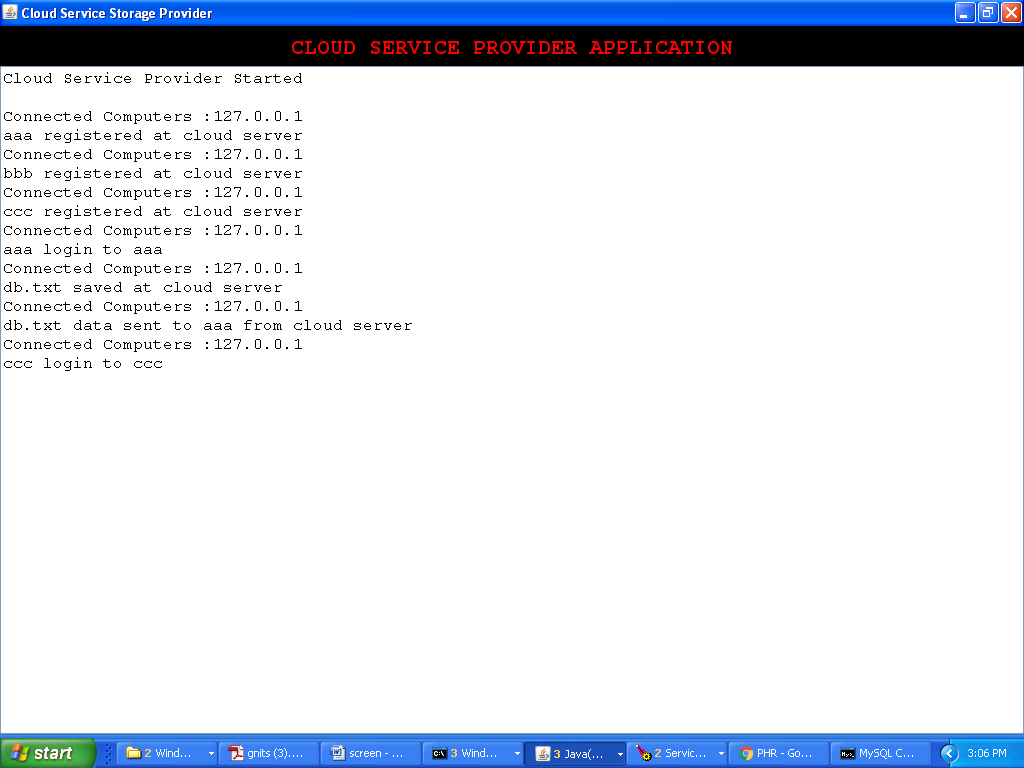
Now click on ‘Click Here’ link to see share data



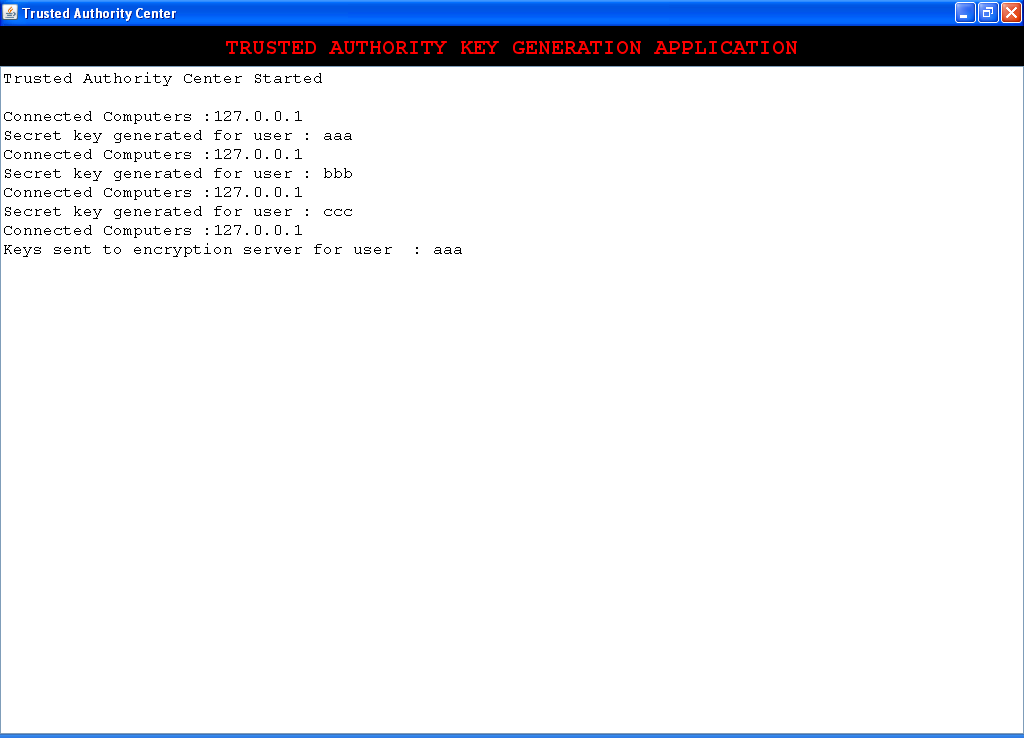
In above screen we can see user aaa has not given access to user ccc. In below screen we can see encryption and decryption server status



Cloud Server screen



Trusted Authority screen



**8. 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.

**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.

[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.