P- 1819

SECURE GROUP COMMUNICATION WITH KERBEROS

A PROJECT REPORT

Submitted by

M.HARI VIGNESH	**71203104010**
V.SATHISH KUMAR	**71203104040**
R.R.NARAIN	**71203104302**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

P- 1819

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE-641 006

ANNA UNIVERSITY: CHENNAI 600 025

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**SECURE GROUP COMMUNICATION WITH KERBEROS**" is the bonafide work of

M.HARI VIGNESH	**71203104010**
V.SATHISH KUMAR	**71203104040**
R.R.NARAIN	**71203104302**

who carried out the project work under my supervision.

[illegible]
SIGNATURE
Dr.S.Thangasamy
HEAD OF THE DEPARTMENT

Dept. of Computer Science & Engineering,
Kumaraguru College of Technology,
Coimbatore-641006.

[illegible]
SIGNATURE
Ms.R.Kalaiselvi
SUPERVISOR
Senior Lecturer

Dept. of Computer Science & Engineering,
Kumaraguru College of Technology,
Coimbatore-641006.

The candidates with University Register Nos. 71203104010, 71203104040, 71203104302 were examined by us in the project viva-voice examination held on 25.04.2007...

[illegible]
INTERNAL EXAMINER

[illegible] 25/4/07
EXTERNAL EXAMINER

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We express our sincere thanks to **Dr.K.Arumugam,** Correspondent, Kumaraguru College of Technology and the management, for providing us this opportunity to undertake this project work.

We express our profound gratitude to **Dr.Joseph .V. Thanikal** Principal, Kumaraguru College of Technology, Coimbatore, for permitting us to undergo a project work.

We are greatly indebted to **Dr.S.Thangasamy,** Professor and Head of the Department of Computer Science and Engineering, for the immense support he has provided us throughout our project.

We extend our sincere thanks to our project coordinator **Ms.S.Rajini,** Senior Lecturer, Department of Computer Science and Engineering, for her constant support and encouragement.

We would like to express our heartfelt thanks to our guide, **Ms.R.Kalaiselvi**, Senior Lecturer, Department of Computer Science and Engineering, who gave the initial idea and guidance to carry out every further step and for her everlasting counseling and untiring help throughout our project.

We sincerely thank our class advisor, **Mr.S.Mohanavel**, Senior Lecturer, Department of Computer Science and Engineering, for his unconditional support and motivation in helping us in completing the project.

We would also like to thank all the faculty members and lab technicians of Department of Computer Science and Engineering for their valuable guidance, support and encouragement during the course of project work.

DECLARATION

DECLARATION

We,

M.HARI VIGNESH	**71203104010**
V.SATHISH KUMAR	**71203104040**
R.R.NARAIN	**71203104302**

Declare that the project entitled "**SECURE GROUP COMMUNICATION WITH KERBEROS**", submitted in partial fulfillment to Anna University as the project work of Bachelor of Engineering (Computer Science) degree, is a record of original work done by us under the supervision and guidance of Ms.R.Kalaiselvi B.E., Senior Lecturer, Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore.

Place: Coimbatore

Date: 23-4-07

[illegible]
[M.HARI VIGNESH]

V. Sathish Kumar
[V.SATHISH KUMAR]

RR Narain
[R.R.NARAIN]

ABSTRACT

ABSTRACT

Group communication applications are growing rapidly in today's internet world. At the same time security becomes the major concern for group communication application. In this project the authentication is enhanced using 'Kerberos' protocol and the Group key Communication is framed based upon the 'Diffie Hellman' Algorithm.

Kerberos thrives to provide the following security features - It makes sure that a server must be well assured of whom a client trying to connect to it. It also assures that a clients request is sent to the stipulated server and no one else on the network. It secures that the password should not be sent across the network since eaves dropper could capture it and play foul with it.

This project presents a new secure authenticated group key exchange algorithm for large groups. The protocol scales efficiently and performs well for dynamic group operations such as join, leave, merge and sub-grouping. The algorithm converts any underlying two-party key exchange algorithm into an efficient group key exchange algorithm.

Users form a tree structure in order to achieve desirable scalability in communication and avoid computational overhead. The tree structure is formed based upon the number of users authenticated into a particular service, and by the use of Diffie Hellman algorithm the parent key is computed. Likewise the intermediate nodes join together in computing the key of the root node (Group key). The computation of group key is done by all the users in the group individually.

Thus by Incorporating Kerberos protocol into this key management algorithm a better security for today's group communication applications is ensured.

TABLE OF CONTENTS

TABLE OF CONTENTS

CHAPTER NO. TITLE PAGE NO.

TABLE OF CONTENTS

CHAPTER NO. TITLE PAGE NO.

INTRODUCTION

CHAPTER – 1

INTRODUCTION

Modern computer systems provide service to multiple users and require the ability to accurately identify the user making a request. In traditional systems, the user's identity is verified by checking a password typed during login; the system records the identity and uses it to determine what operations may be performed. The process of verifying the user's identity is called authentication. Password based authentication is not suitable for use on computer networks. Passwords sent across the network can be intercepted and subsequently used by eavesdroppers to impersonate the user. While this vulnerability has been long known, it was recently demonstrated on a major scale with the discovery of planted password collecting programs at critical points on the Internet. Stronger authentication methods base on cryptography are required. When using authentication based on cryptography, an attacker listening to the network gains no information that would enable it to falsely claim another's identity. Kerberos is the most commonly used example of this type of authentication technology. Unfortunately, strong authentication technologies are not used as often as they should be, although the situation is gradually improving.

With the emergence of many group-oriented distributed applications such as tele-conferencing, video-conferencing and multiplayer games, there is a need for security services to provide group-oriented communication privacy and data integrity. To provide this form of group communication privacy, it is important that members of the group can establish a common secret key for encrypting group communication data. To illustrate the utility of this type of applications, consider a group of people in a peer-to-peer or

have a previously agreed upon common secret key, communication between group members is susceptible to eavesdropping.

The undertaken project provides a means to communicate secure data among the members of a dynamic peer group using Tree-based Diffie-Hellman approach. Also, it provides a means to effectively manage the group key, using interval-based rekeying algorithms. The interval-based algorithms recompute the group key in specific rekeying intervals and thereby reducing the communication overhead and the computation costs.

BASICS OF 'GROUP KEY COMMUNICATION WITH KERBEROS'

CHAPTER – 2

BASICS OF 'GROUP KEY COMMUNICATION WITH KERBEROS'

2.1. Kerberos – Authentication

In the Kerberos System, a series of messages is exchanged between principals and the authentication server, as well as between the principals themselves (the client and server). Tickets must be obtained from the authentication server and then exchanged between the client and server to perform authentication.

Kerberos Authentication System has various stages of communication as follows:

- ❖ Communication between the Client and the Authentication Server (AS), also know as the Key Distribution Center (KDC).
- ❖ Communication between the Client and the Ticket Granting Server (TGS)
- ❖ Communication procedures between the Client and the Application Server.

2.1.1 The Authentication Service (AS) Exchange

The first exchange of messages takes place between a client and the Authentication Service (KDC). The exchange is used to obtain a ticket and session key for use with a server when no credentials were previously obtained. This exchange is typically initiated by a client, usually at the start of a login session, to obtain credentials for the Ticket Granting Service, which can then be used to obtain tickets for other servers. In the case of a this exchange is also used to obtain a ticket for such a server. This exchange consists of two messages, the first being a request from the client to the KDC in which it specifies the credentials it wants and also certain options. The second is the reply from the KDC containing the ticket and the session key to use. The clients secret key (usually derived from a password) issued for encryption.

The first message, sent from the client to the KDC, is the KRB_AS_REQ message. It consists of the following components

KRB AS REQ ={pvno, msgtype, [padata], reqbody}

reqbody =

{kdcoptions, [cname], realm, [sname], [from], till,

[rtime], nonce, [etype], [addresses . . .]}

The client presents this message to the server in unencrypted form identifying itself, and providing the server for which the ticket is meant (typically the Ticket-Granting Server) as well as the nonce that prevents replay.

The authentication server, upon receiving this message, will look up the client and server in its database, and get the secret key for both of them. It will also generate the random session key. It prepares the ticket using the options specified in the KRB_AS_REQ message and encrypts them using the client's key (Password).

It will then send a message to the client, the KRB_AS_REP message.

KRB AS REP =

{pvno, msgtype, [padata], crealm, cname, {ticket}Ks, {encpart}Kc}

encpart =

{key, lastreq, nonce, keyexpiration, flags, authtime, [starttime],

endtime, [renewtill], srealm, sname, [caddr]}

After client receives the response, it will try to decrypt the message. Upon successful decryption, it can now use the ticket to contact the Ticket Granting Server, as described in the following section.

2.1.2 The Ticket Granting Service (TGS) Exchange

As mentioned previously, the AS Exchange is used primarily to get a ticket for a special server, the Ticket Granting Server (TGS), which can be used to obtain additional tickets without having to use the client's secret key anywhere in the process. This protects the secrecy of the client's secret, and makes Kerberos more transparent to the user. If this had not been done, then any time a ticket needs to be requested the client's secret key would be needed. This would mean that either the user has to enter their password every time (which is a nuisance), or the client's workstation needs to cache the secret key. Since Kerberos does not assume that the workstations are secure, this compromises the security of the secret key. Therefore, this additional ticket exchange is used.

The TGS itself needs to have the access to all the secret keys. Although it does not encrypt the reply using the client's secret key, it still needs to encrypt the ticket using the server's secret key. Therefore, the TGS is often the same physical system as the KDC.

The TGS Exchange is much like the AS Exchange. Here too, two messages are sent, one from the client to the TGS that requests a ticket, and one from the TGS.

These two messages as KRB_TGS_REQ and KRB_TGS_REP. These two messages are in structure very similar to the KRB_AS_REQ and

protocol, both these messages are described to be instances of the KRB_KDC_REQ and KRB_KDC_REP messages respectively.

The first, and perhaps most important, difference is that in the TGS Exchange the encryption is not done using the client's secret key, as was mentioned before. Instead, the encpart section of the response is encrypted using the session key that is a part of the the ticket-granting ticket, which was previously distributed in the AS Exchange. Alternatively, a sub session key can be used that is specified in the authenticator for the message.

Once the TGS has received the message, it will verify that it is valid and that the TGT held within is valid and has not expired. If all checks out, the reply is sent; once again much like was done in the AS Exchange. Several pieces of data, such as the client's name and addresses are by default copied from the TGT, although these can be different for forwarded or proxy tickets. Once the client has received the reply, it can once more check if the ticket is usable, and then use it to contact the destination server.

2.1.3 The Client/Server (CS) Exchange

The client, through all these previous exchanges, had really only one goal in mind. It wanted to contact a server. Unfortunately, this server had decided it needed authentication, causing the client to first go through the first two exchanges at least once. Of course, once a ticket has been obtained for a server, and it has not expired yet, the AS and TGS Exchange can be skipped and the client can immediately move forward to the last exchange, the Client/Server Exchange.

In this exchange, only a single message is required, that is from the client to the server. Optionally, the server can also send a reply message the client as well. The client generates the KRB_AP_REQ message to the server, sometimes also called the authentication header after having acquired a ticket for use with the server.

This message has the following format:

KRB AP REQ =

{pvno, msgtype, apoptions, ticket, authenticator}

Once the KRB_AP_REQ message is received by the server, it is checked for validity. The message is valid if the ticket contained in it can be decrypted using the server's secret key, and if the authenticator can be decrypted using the session key contained in the ticket. The name and realm of the client specified in the ticket are validated against the ones specified in the authenticator. Of course, the ticket may not be expired, and the authenticator's time may not vary too much from the server's local time. Within the allowed time skew, a server must remember all the authenticators it received to prevent replay attacks. If the message checks out, the server is ensured of the client's identity. Usually, the client will send its initial request (the request it needed to authenticate for) together with the KRB_AP_REQ message.

2.2. Group Key Communication

This Project uses the TGDH (Tree-Based Group Diffie-Hellman) protocol to effectively maintain the group key in a dynamic peer group. Also, the proposed approach uses interval-based rekeying techniques for recomputing the group keys on various membership events. The interval-based techniques reduce the communication overhead and the computation costs in the recomputation of the group key.

2.2.1. Tree-Based Group Diffie-Hellman Protocol

The proposed approach uses the TGDH (Tree-Based Group Diffie-Hellman) protocol to effectively maintain the group key in a dynamic peer group. Each member maintains a set of keys and is arranged in a hierarchical binary tree. Every tree node is assigned a node ID (v). Each node is associated with a secret key (K_v) and a public key (BK_v). The public key is generated by

$$BK_v = a^{Kv} \bmod p$$

Each leaf node corresponds to the individual secret and public keys of the member Mi. Each member holds all the secret keys along its key-path. Each non-leaf node has two children whose ID's are given by 2v+1 and 2v+2. The secret key of a non-leaf node is computed using the secret key of one child and the public key of the other child.

$$K_v = a^{K2v+1K2v+2}$$

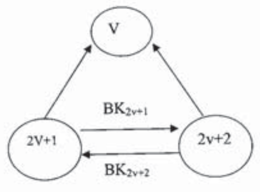


Fig 4.1 Secret key of a non-leaf node

Diffie-Hellman

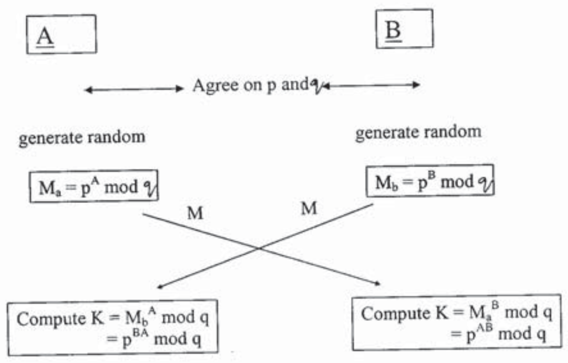


Fig 2.2.1 Basic of Diffie Hellman Algorithm

The secret key of a non-leaf node v can be generated by :

$$K_v = (BK_{2v+1})\, K_{2v+2} \bmod p = (a^{K2v+1})\, K_{2v+2} \bmod p.$$
$$K_v = (BK_{2v+2})\, K_{2v+1} \bmod p = (a^{K2v+2})\, K_{2v+1} \bmod p.$$

ANALYSIS OF THE PROBLEM

CHAPTER – 3

ANALYSIS OF THE PROBLEM

The proposed system has to be analysed well for its merits and demerits. Our main aim is to make use of all its merits in a particular problem and to remove the demerits of it. First the proposed system is analyzed then the merits of the system are discussed.

The requirement specification of the proposed system should also be analysed so that the hardware and software requirement of the proposed system is known in order to execute our application. This helps the user in developing his application.

The survey of the proposed system should be made in order to find whether any software is needed to develop our application, also it has been found whether there is any benefit to the system when using that particular software. And also it is analysed whether this software will be useful to user.

The documentation of the proposed system is made which helps us to collect the details that are describing the system. And the graphical representation of the system and its activities are presented in order to make the user understand the system. Then the records and description of the system elements are analyzed and maintained.

3.1 Problem Definition

The purpose of this project is to provide an efficient key management technique and the secure authentication technique which can be used for securely sharing the keys and which is efficient and scalable after authenticating the use in a secure way.

The popularity of communication within the Local Area Network (LAN) has recently accelerated along with the risks associated with its use. One of the most obvious risks is the situation where a client tries to gain access to resources that do not belong to him from the server. Then another main trouble is that main server is having burden of keeping all user's login name and password. Because when each time user try's to connect to server login name and password will translate each time in network. So illegal user will have enough time to decrypt user login and password, and gain access to main server.

In Group Key Management, the problem is to first study the various key management techniques currently available in wired networks. The emphasis is on security and efficiency of the key management technique. The goal is to improve the features provided by previous key management techniques, in particular, Basic key management techniques. This dissertation addresses the key distribution problems associated with maintaining communication integrity in the presence of membership changes. We consider the single sender - multiple receiver models of the multicast communications.

3.2. System Analysis

The existing system has to be analyzed well for its merits and demerits. Our main aim is to make use of all its merits and to remove the demerits of it. First we must analyze the existing system and we have to find the drawbacks and difficulties we face by using this system. And proposed the new system then analyze the system and find out whether it satisfies the requirements. And see the advantages of the proposed system and the fields where this particular system can be used, then discuss about the merits of proposed system over the existing system. The process of analyzing a software item to detect the differences between existing and required conditions, and to evaluate the features of the software item.

3.2.1 Existing System

In existing system, for the authentication of the users the main server takes care of the entire job like retrieving user names, passwords across the network securely free from the hackers, and informing the users whether they are authenticated or not by comparing the login details given by the user with the database presented in the system.

In existing system, server keeps a database for storing all the information about the users. So authentication part is carried by main server itself. And group communication is also done by main server itself. In existing system group communication is done by DIFFIE HELLMAN algorithm.

3.2.2 Drawbacks of the Existing System

Since authentication part is done by main server, complexity of server increases if there is more number of users linked into the system. The encrypting and decrypting part will be take care by the server along with providing the services which also creates overhead to the main server.

In existing system, server keeps a database for storing all the information about the users which leads to storage complexity and computational complexity for the group communication. Storage Complexity results in abundant storage of user names, passwords and computational complexity arises when a new key should be derived and distributed across the network for every new user joining, user leaving.

3.2.3 Proposed System

One of the main problems of existing system is authentication part. This authentication part is done by third party server called '*Kerberos Authentication Protocol*'.

Features of Kerberos:

- ❖ That a server must be well assured of whom a client trying to connect to it says it is.
- ❖ A client must be rest assured that his/her request is sent to the stipulated server and no one else on the network.
- ❖ That password should not be sent across the network since eaves dropper could capture it and play foul with it.

Another main advantage of proposal system in Group Communication is that dynamic nature of the group is solved on the basis of 'Interval' Algorithm.

The system proposed a real time solution application for group communication over internet. Proposed system will reduce main server complexity. In group communication part two main algorithms are implemented they are Diffie Hellman and Interval Based Algorithm both collectively known as '*Queue Batch Algorithm*'.

The main advantage of Interval Based Approach is

- ❖ Every time a new user joins a group or leaves a group, rekeying is done.
- ❖ This contributes for authentication.
- ❖ To solve the Dynamic Nature of the group key, we go for interval based algorithm.

DESIGN AND IMPLEMENTATION

CHAPTER – 4

DESIGN AND IMPLEMENTATION

4.1. Data Flow diagram

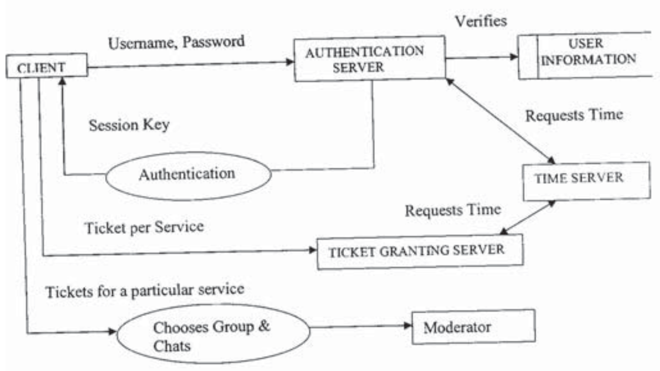


Fig 4.1 Data Flow Diagram

4.2. Detailed Design

1. Authentication Server:

This module concentrates on validating the authentication of user. The following services will be taken cared by this server.

- ❖ Maintains the database of username and passwords.
- ❖ Validates the authentication of the user.
- ❖ If the user is valid, then the session ticket is generated to client.
- ❖ Contacts with the Time server to synchronize the timings from the client.

2. Ticket Generating Server:

This Server does the following activities:

- ❖ Generates ticket to client such that the client contacts with the server.
- ❖ Contacts with the Time server to synchronize the timings from the client which is presented in the ticket.

3. Moderator:

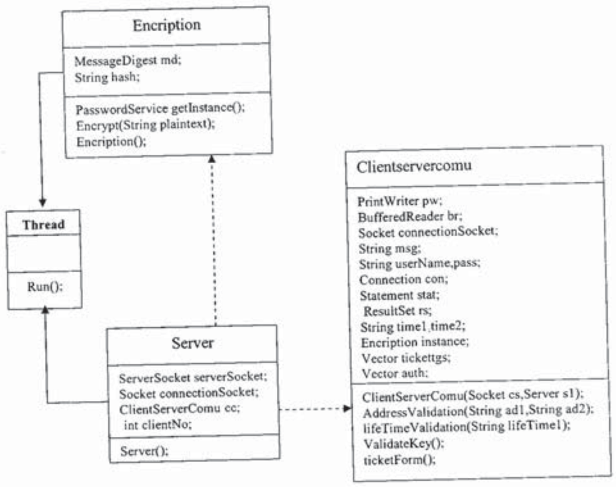
This server provides the service which the client requests. In this Project this moderator provides the chat application for the various chat rooms to provide secure group communication between clients. This implements the rekeying process for every group after a specified amount of time for security purpose.

4. Time Server:

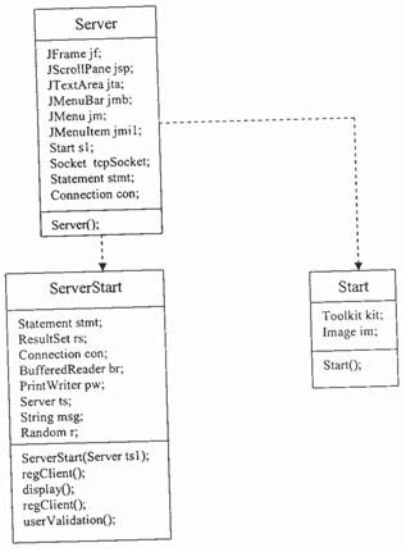
This Time server responses the current time to Ticket synchronization in time for various transactions. This will ensure the security of the session tickets and the service tickets such that no intruders come and re modify the expired ticket and use it.

4.3. Structural Design

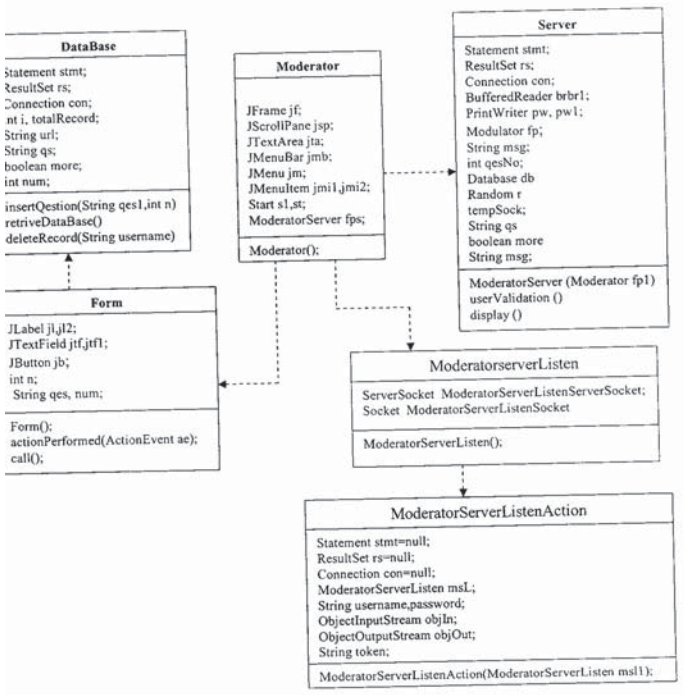
4.3.1. Class Diagrams:



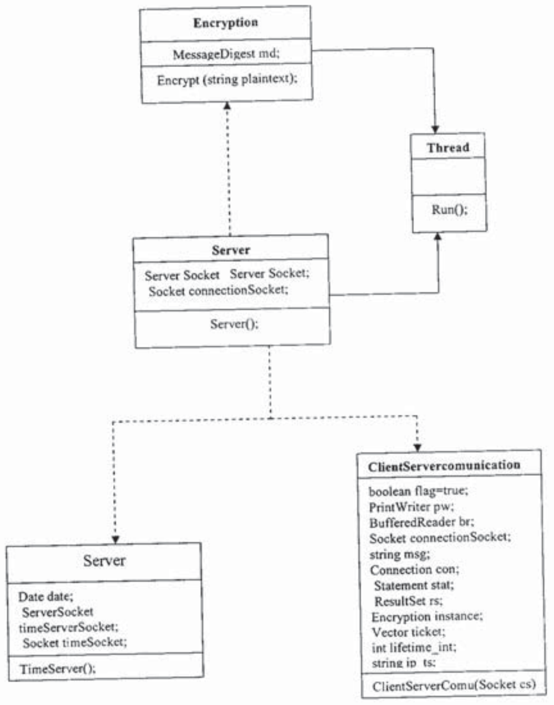
4.3.1.1. TGS Server Class Diagram



4.3.1.2. Third Party Server Class Diagram



4.3.1.3. Moderator Server Class Diagram



4.3.1.4. Authentication Server Class Diagram

4.4. Requirement Specification

4.4.1. Hardware Requirement

❖	Processor	Intel Pentium III
❖	CPU	1.7 GHz.
❖	RAM	64 MB
❖	Monitor	15" Samsung color monitor
❖	Keyboard	Standard Keyboard with 104 Keys
❖	Mouse	Serial Mouse

4.4.2 Software Requirement

- ❖ JDK 1.4 or more
- ❖ Operating Systems: Microsoft Windows 98, NT4 SP3, 2000, Me or XP.

4.4.3 Features of the Software Used

The key that allows Java to solve both the security and portability problems is that the output of a Java complier is not executable code. Rather it is Bytecode. Bytecode is a highly optimized set of instructions designed to be executed by the Java runtime system, called the Java Virtual Machine(JVM).

Translating a java program into bytecode helps make it much easier to run a program in a variety of environments. The fundamental forces that necessitated the invention of Java were security and portability. The key considerations were summed up by the java team in the following

- ❖ Simple
- ❖ Secure
- ❖ Portable
- ❖ Object-oriented
- ❖ Robust
- ❖ Multithreaded
- ❖ Architecture-neural
- ❖ High performance

4.5. Implementation

In the same way as the communication between the Client and the AS, in this exchange, both the Client and the Ticket Granting Sever must get the time from the Server, thereby making it impossible for users to alter the time during the exchange. Except for the time, the other procedures are same as that of the Client/TGS communication of the Kerberos.

The interval-based rekeying algorithms performs recomputation of group key over a batch of join or leave events at specific intervals. The rebuild and batch algorithm perform all the updates at the beginning of the rekeying interval. This necessitates the queue-batch algorithm, which works in two phases. 1. Queue-sub tree phase: The queue-batch algorithm outperforms these algorithms by utilizing the elapsed interval in constructing the temporary tree with the new members joined during the idle interval and renewing the nodes along the temporary key path.

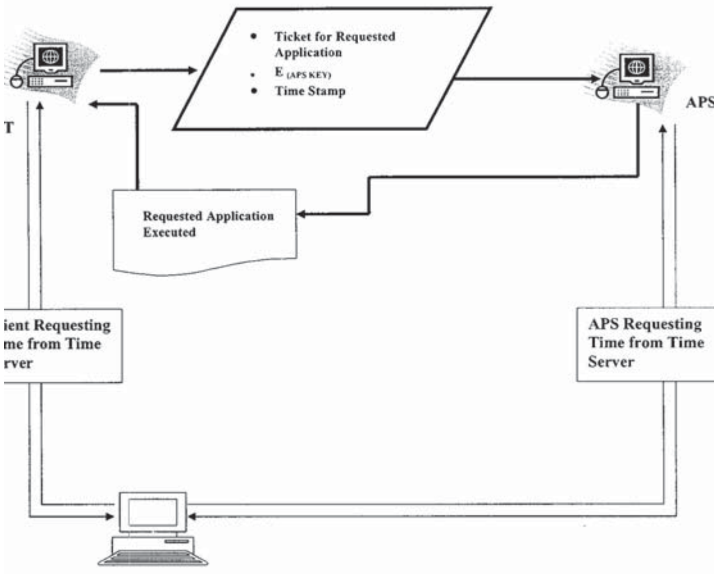

Ticket for Requested Application
E (APS KEY)
Time Stamp
T
APS
Requested Application Executed
ient Requesting me from Time rver
APS Requesting Time from Time Server
Time Server

Fig.4.5.1 Architecture Diagram of Kerberos

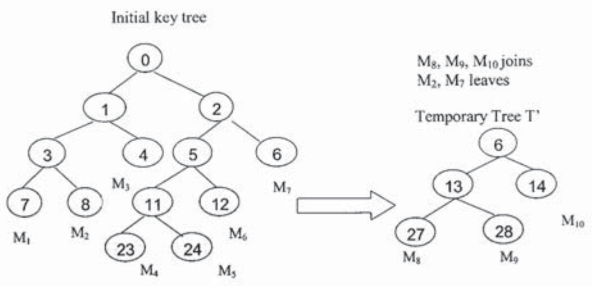


Fig.4.5.2. Tree structure of users in a particular group

The sub-tree is merged with the original tree and the group key is recomputed by renewing the nodes along the key path. In this case, we append this new member in a temporary key tree T'. The second phase occurs at the beginning of every rekeying interval and we merge the temporary tree T' (which contains all newly joining members) to the existing key tree T.

The Queue-batch algorithm is illustrated in Fig. 11, where members M_8, M_9 and M_{10} wish to join the communication group, while M_2 and M_7 wish to leave. Then the rekeying process is as follows: (i) In the *Queue-sub tree* phase, the three new members M_8, M_9 and M_{10} first form a tree T'. In this case, M_{10} is elected to be the sponsor. (ii) In the *Queue-merge* phase, the tree is added at the highest departed position, which is at node 6. Also, the blinded key of the root node of T', which is BK_6, is broadcast by M_{10}. (iii) The sponsors M_1, M_6 and M_{10} are elected. M_1 renews the secret key $K_1$and broadcasts the blinded key BK_1. M_6 renews the secret key K_2 and broadcasts the blinded key BK_2. (iv) All members can compute the group key.

TESTING

CHAPTER – 5

TESTING

The testing process focuses on the logical internals of the software, ensuring that all statements have been tested, and on the functional externals; Implementation or System Testing is the stage where the theoretical Design is converted into a Working System. This stage consists of the following steps.

- ❖ Making necessary changes to the system as desired by the user.
- ❖ Training the user personal. Prior to the Implementation of two stages shown below has been carried out
- ❖ Testing the developed programs with the sample data.
- ❖ Detecting and correcting the errors.

Testing is a set of activities that can be planned in advance and conducted systematically. For this reason a template for software testing which is a set of steps into which we can place specific test case design techniques and testing methods-should be defined for the software process.

A number of software testing strategies have been proposed in the literature. All provide the software developer with the template for testing and all have the following generic characteristics:

- ❖ Testing begins at the component level and works outward toward the integration of the entire computer-based system.
- ❖ Different testing techniques are appropriate at different points in time.
- ❖ The developer of the software and an independent test group conducts testing.
- ❖ Testing and debugging are different activities, but debugging must be accommodated in any testing strategy.

A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements. Some of the testing which is applicable for this project are tested and presented below.

5.1. Unit testing

Unit testing concentrates on each unit of the software as implemented in the source code. It focuses verification effort on the smallest unit of software design-the component or module. Using the component-level design description as a guide, important control paths are tested to uncover errors within the boundary of module.

In this testing the developed modules are tested when they are created and errors are rectified in each module. Sample data are used for testing case is taken in providing the sample data.

The module interface is tested to ensure if the information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.

Unit testing can help during the initial development of the class or module's public interface. Unit tests force to think about how another developer will want to use the code while writing that code. This shift in focus can help to present a smaller, cleaner interface to the classes and modules. This benefit is most often associated with test-driven development. For that in the application that is developed many modules have been used,

The two test sets for GUI based tests are:

Test Set 1: For Windows

1. Are all functions that relate to the window operational?
2. Are all relevant, controls, dialog boxes, and buttons available and properly displayed for the window?

Test Set 1(results):

1. Yes, all functions relate to that window.
2. Yes, all controls & objects are displayed at their appropriate places.

Test Set 2: Mouse Operations

1. Are all functions properly addressable by the mouse?
2. Does each function perform as advertised?
3. Do multiple or incorrect mouse picks within the window caused unexpected side effects?

Test Set 2(results):

1. Yes, all are addressable by the mouse.
2. Yes , function perform as advertised.
3. No, incorrect mouse picks within the window do not generate unexpected side effects.

5.2. Integration testing:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design.

The program is constructed and tested in small increments, where errors are easier to isolate and correct; interfaces are more likely to be tested completely; and a systematic test approach may be applied. Incremental integration tests like top down integration and bottom up integration test is tested and the errors are rectified. Integration test document contains the test plan that describes the overall strategy for integration.

And here testing is divided into phases and builds that address specific functional and behavioral characteristics of the software. As module is successfully unit tested an integrated test is done to incorporate each module into overall software structure.

The Integration testing done when the following modules are integrated:

- ❖ Authentication Server
- ❖ Ticket Granting Server
- ❖ Time Server
- ❖ Third Party Server
- ❖ Moderator

5.3. User testing

In user testing the user himself will test the software with some sample data's and check whether any errors occur and if so he will try to rectify it, and give us error free software. So the software that is developed will work efficiently in all circumstances. The system group whose suggestions are incorporated to form the overall system tests the developed prototype of the project.

5.4. Reliability Testing

This is to check that the server is rugged and reliable and can handle the failure of any of the components involved in providing the application. In case of failure of the server, the intimation is done to the client that the server is currently unavailable for service.

CONCLUSION AND FUTURE SCOPE

CHAPTER – 6

CONCLUSION AND FURTHER SCOPE

In conclusion, this project is not touching on all the features of the Kerberos Protocol, but rather it satisfies one of the limitations associated with this Protocol and tried to seek a more refined way of going around the limitation.

In the future scope of this project we seek to bring it to a higher level. We seek to enhance the Kerberos Protocol in the following ways:

- ❖ To have more than one Authentication Server and Ticket Granting Server. This will reduce the burden on these servers. You can imagine about 20 clients in an organization trying to connect to a server all at the same time. This may slow down the response time of the server no matter the configuration of the computer. But if there is more than one server doing the same job then at any point in time, the client model will look out for the one that has few or no tasks to communicate to. This will increase the response speed.
- ❖ That all servers must authenticate themselves to the timeserver too. If this is done, there can be a log which will keep track of all computers requesting time from this server at time. This can be used for future record or something of that sort.
- ❖ The system that was developed provides a good means for communicating the information among the group members in a secure manner. The threshold-based rekeying technique, when integrated with the interval-based rekeying algorithm-Queue-Batch reduces the computation overhead and the communication costs in recomputing

APPENDIX

APPENDIX 1

1. Sample code:

Module 1: Authentication server

1.1. ClientServerComu:

```
import java.net.*;
import java.io.*;
import java.sql.*;
import java.util.Date;
import java.util.StringTokenizer;
import java.util.Vector;
import java.text.*;

class ClientServerComu extends Thread
{
        boolean flag=true;
        PrintWriter pw;
        BufferedReader br;
        Socket connectionSocket;
        String msg;
        String  userName,pass,timestamp,ec_key,etgs_key,tgsname,
                clientAddress;
        Connection con;
        Statement stat;
        ResultSet rs;
        String time1,time2,lifeTime="3";
        Encription instance;
        Vector ticket=new Vector();
        int lifetime_int=5;// lifetime to login into  Ticket Granting Server
        String ip_ts="90.0.0.44";
        ClientServerComu(Socket cs)throws Exception
        {
          super("ClientServerComu");
          this.connectionSocket=cs;
          instance = new Encription();
        pw=new
      PrintWriter(connectionSocket.getOutputStream(),true);
      br=new BufferedReader(new
      InputStreamReader(connectionSocket.getInputStream()));
      start();
      }

      public void run()
      {
      try
      {
             while(flag)
      {
             msg=br.readLine();
             if(msg.equals("AES"))
             {
                    String detail=br.readLine();
    //              System.out.println("hi");
                    System.out.println("detail = "+detail);// detail =
      test/TGS1/TGS1/Sun Apr 15
                    StringTokenizer st=new StringTokenizer(detail,"/");
                           userName=st.nextToken();
                           pass=st.nextToken();
                                  tgsname=st.nextToken();
                                  time1=st.nextToken();
                    ec_key=instance.encrypt(pass);// Encryption for p
      password
                    etgs_key=instance.encrypt(userName+tgsname);
                    System.out.println("\n\n\n----------------Authentication
Server Exchange ------------------\n\n");
                    System.out.println("\n\n (2) AS -> C :
E_K_c||K_c,tgs||ID_tgs||TS_2||Lifetime_2||Ticket_tgs\n\n");
                    System.out.println("   E_K_c : "+ec_key+"\n");
                    System.out.println("   K_c,tgs : "+etgs_key+"\n");
                    System.out.println("   ID_tgs : "+tgsname+"\n");
                    try
                           {
                                  Socket s=new Socket(ip_ts,2500);
                                  BufferedReader br=new
BufferedReader(new InputStreamReader(s.getInputStream()));
```

```
                                 }
                                 catch(Exception e)
                                 {
                                               System.out.println("Error while
connecting Time Server : "+e);
                                        }

                                        System.out.println("   TS_2 : "+time2+"\n");
                    System.out.println("   Lifetime_2 : "+lifetime_int+"
minutes \n");
             pw.println(ec_key);
             pw.println(etgs_key);
             pw.println(tgsname);
             pw.println(time2);
             SimpleDateFormat sdf=new SimpleDateFormat("EEE MMM d
h:m:s a yyyy ");
             Date d1=(Date)sdf.parseObject(time2);
             d1.setMinutes(d1.getMinutes()+lifetime_int); // Deadline time
for the ticket to be sent in the ticket
             lifeTime=sdf.format(d1);
             pw.println(lifeTime);
             ticket.addElement(etgs_key);
             ticket.addElement(ec_key);
             ticket.addElement(userName);

        clientAddress=connectionSocket.getRemoteSocketAddress().toString(
);
          // System.out.println("Client Address = "+clientAddress);
             String[] temp=clientAddress.split(":"); //result is
/127.0.0.1:1032
             temp=temp[0].split("/");
             ticket.addElement(temp[1]);
             ticket.addElement(tgsname);
             ticket.addElement(lifeTime);

             pw.println(ticket.elementAt(0));
             pw.println(ticket.elementAt(1));
             pw.println(ticket.elementAt(2));
             pw.println(ticket.elementAt(3));
             pw.println(ticket.elementAt(4));
```

```
             //ObjectOutputStream oos=new
ObjectOutputStream(connectionSocket.getOutputStream());
             //oos.writeObject(ticket);
             //oos.flush();
             System.out.println("    Elements at Ticket_tgs : ");
                                 System.out.println("
"+ticket.elementAt(0));
                                 System.out.println("
"+ticket.elementAt(1));
                    System.out.println("      "+ticket.elementAt(2));
                    System.out.println("      "+ticket.elementAt(3));
                    System.out.println("      "+ticket.elementAt(4));
                    System.out.println("      "+ticket.elementAt(5));
             pw.println("GETIT");
             flag=false;
             connectionSocket.close();
             }
      }
      }
      catch(Exception e)
      {
             System.out.println("Error: ClientServerComu Classs Method
:run "+e);
      }
      }
 }
```

1.2. Encryption:

```
                    import java.security.MessageDigest;
 import java.security.NoSuchAlgorithmException;

 import sun.misc.BASE64Encoder;
 import sun.misc.CharacterEncoder;

 public final class Encription
 {
  private static  Encription instance;

  public Encription()
```

```
 }

 public synchronized String encrypt(String plaintext) throws Exception
 {
  MessageDigest md = null;
  try
  {
   md = MessageDigest.getInstance("SHA"); //step 2
  }
  catch(Exception e)
  {
   throw new Exception(e.getMessage());
  }
  try
  {
   md.update(plaintext.getBytes("UTF-8")); //step 3
  }
  catch(Exception e)
  {
   throw new Exception(e.getMessage());
  }

  byte raw[] = md.digest(); //step 4
  String hash = (new BASE64Encoder()).encode(raw); //step 5
  return hash; //step 6
 }

/*  public static synchronized PasswordService getInstance() throws
Exception//step 1
  {
   if(instance == null)
   {
   }
   return instance;
  }
*/

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

1.3. Server:

```
 import java.net.*;
 import java.io.*;
 import java.util.*;

public class  Server  extends Thread
{
        ServerSocket serverSocket;
        Socket connectionSocket;

        Server()throws Exception
        {
                super("Server");
                serverSocket=new ServerSocket(10500);
                start();
        //Runs infinetly
        }

        public void run()
        {
        try
        {
                while(true)
                {
                        connectionSocket=serverSocket.accept();     // Accepts
 any ServerSocket type from client and converts into local socket
                        new ClientServerComu(connectionSocket);
                }
        }
        catch(Exception  e)
        {
                System.out.println("Error: ServerClasss Method :run "+e);
        }
        }

        public static void main(String [] a) throws Exception
        {
                new Server();
        }
}
```

Module 2: TGS Server

2.1. ClientServerComu:

```
import java.net.*;
import java.io.*;
import java.sql.*;
import java.util.Date;
import java.util.StringTokenizer;
import java.util.Vector;
import java.text.*;

class ClientServerComu extends Thread
{
        PrintWriter pw;
        BufferedReader br;
        Socket connectionSocket;
        String msg;
        String
userName,pass,timestamp,ec_key,etgs_key,tgsname,clientAddress,ev_key,e
cv_key,serverName,time4,lifeTime4;
        Connection con;
        public  ObjectInputStream oos;
        Statement stat;
        ResultSet rs;
        String time1,time2,lifeTime="3";
        Encription instance;
        Vector tickettgs;
        Vector auth;
        Vector ticketV=new Vector();
        Server s;
        int cl_no=0;
        Date d2;

        public static String ip_ts="90.0.0.44";

        public ClientServerComu(Socket cs,Server s1)throws Exception
        {
        super("ClientServerComu");
        s=s1;
           instance = new Encription();
           this.connectionSocket=cs;
        instance = new Encription();
        pw=new PrintWriter(connectionSocket.getOutputStream(),true);
        br=new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
        start();
        }
```

2.2 Encryption:

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

import sun.misc.BASE64Encoder;
import sun.misc.CharacterEncoder;

public final class Encription
{
 private static  Encription instance;

 public Encription()
 {
 }

 public synchronized String encrypt(String plaintext) throws Exception
 {
  MessageDigest md = null;
  try
  {
   md = MessageDigest.getInstance("SHA"); //step 2
  }
  catch(Exception e)
  {
   throw new Exception(e.getMessage());
  }
  try
  {
   md.update(plaintext.getBytes("UTF8")); //step 3
  }
  catch(Exception e)
  {
   throw new Exception(e.getMessage());
  }

  byte raw[] = md.digest(); //step 4
  String hash = (new BASE64Encoder()).encode(raw); //step 5
  return hash; //step 6
 }

/*  public static synchronized PasswordService getInstance() throws
Exception//step 1
 {
  if(instance == null)
  {
  }
  return instance;
 }
*/

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

        instance = new Encription();
    }
}
```

2.3. Server

```
import java.net.*;
import java.io.*;
import java.util.*;
```

```
public class  Server  extends Thread
{
        ServerSocket serverSocket;
        Socket connectionSocket;
        ClientServerComu cc;
        int clientNo=0;

        Server()throws Exception
        {
                super("T G Server");
                serverSocket=new ServerSocket(3500);
                start();
        }

        public void run()
        {
                try
        {
                while(true)
                {
                        connectionSocket=serverSocket.accept();
                        clientNo++;
                        cc=new ClientServerComu(connectionSocket,this);
                }
        }
        catch(Exception  e)
        {
                System.out.println("Error: ServerClasss Method :run "+e);
        }
        }

        public static void main(String [] a) throws Exception
        {
                new Server();
        }
  }
```

2.4. TgsSerialized

```
import java.io.*;
```

```
class TgsSerialized implements Serializable
{

 Vector tgsTicket;
 Vector auth;
 TgsSerialized()
 {

 }
 public void setTgsTicket(Vector v)
 {
 tgsTicket=v;
 }
 public Vector getTgsTicket()
 {
  return tgsTicket;
 }
 public void setAuthTicket(Vector v)
 {
  auth=v;
 }

 public Vector getauthTicket()
 {
  return auth;
 }

 }
```

Module 3: Third Party Server

3.1. Server

```
import java.net.*;
import java.io.*;
import javax.swing.*;
import  java.awt.event.*;
```

```
class  Server extends Thread
{
        JFrame jf;
        JScrollPane jsp;
        JTextArea jta;
        JMenuBar jmb;
        JMenu jm;
        JMenuItem jmi1;
        Start s1;
        ServerSocket  tcpServerSocket=null;
        Socket  tcpSocket=null;
        ServerStart tss;
        Statement stmt=null;
    ResultSet rs=null;
        Connection con=null;

        public Server()throws Exception
        {
                        super("Third party Server");
        //      System.out.println("test");

        //      new SendFile();
                jf=new JFrame("Server");
        s1=new Start();
        jf.getContentPane().setLayout(null);
        jf.setBounds(50,50,600,400);
        jta=new JTextArea();
        jta.setText("           SERVER                 ");
        jta.setEditable(false);
        jsp=new JScrollPane(jta);
        jsp.setBounds(10,10,570,320);
        jf.getContentPane().add(jsp);
        jmb=new JMenuBar();
        jf.setJMenuBar(jmb);
        jm=new JMenu("Help");
        jmi1=new JMenuItem("About");
        jm.add(jmi1);

                jmi1.addActionListener(new ActionListener()
                public void  actionPerformed(ActionEvent ae)
                {
                        s1.setVisible(true);
                }
            });

            jmb.add(jm);
        jf.setVisible(true);
        jf.setDefaultCloseOperation(3);
                tcpServerSocket=new ServerSocket(1500);
                start();
        }

        public void run()
        {
                while(true)
        {
                try
                {
                tcpSocket=tcpServerSocket.accept();
                        tss=new ServerStart(this);
                        }
                        catch(Exception e)
                {
                }
        }
        }

        public static void main(String[] s)throws Exception
        {
        new Server();
        }
 }
```

3.2. ServerStart:

```
import java.net.*;
import java.io.*;
import java.sql.*;
import java.util.*;
```

```
mport java.text.*;

:lass  ServerStart extends Thread
{
      Statement stmt=null;
  ResultSet rs=null;
      Connection con=null;
  BufferedReader br=null;
      PrintWriter pw;
      Server ts;
      String msg;
      String
username="",password="",sex="",age="",ans1="",ans2="",ans3="",ans4="",
ans5="",usernamet="";
      String[] qestion=new String[100];
      int qesNo=0;
      Random r;
      SimpleDateFormat sdf=new SimpleDateFormat("hh:mm:ss a");
      String fileName="";

      public  ServerStart(Server ts1)throws Exception
      {
             super("Third party Server");
      ts=ts1;
      pw=new PrintWriter(ts.tcpSocket.getOutputStream(),true);
             br=new  BufferedReader(new
 InputStreamReader(ts.tcpSocket.getInputStream()));
             r=new Random();
             start();
      }

      public void run()
      {
             String msg;
             try
      {
             while(true)
             {
                    msg=br.readLine();
                    System.out.println(msg);
```

```
             {
             System.out.println("System send info to client");
             FileReader fw=new FileReader("group.ini");
             BufferedReader br1=new BufferedReader(fw);
             String temp=br1.readLine();
             while(temp!=null)
             {
                    System.out.println(temp);
                    pw.println(temp.split(":")[0]);
                           temp=br1.readLine();
             }
             pw.println("END");
                    fw.close();
             }
             else if(msg.equals("GDE"))
             {
             String groupName=br.readLine();
                           FileReader fw=new FileReader("group.ini");
             BufferedReader br1=new BufferedReader(fw);
             String temp=br1.readLine();
             while(temp!=null)
             {
                    System.out.println(temp);
                    if(temp.split(":")[0].equals(groupName))
                    {
                           pw.println(temp.split(":")[1]);
                           pw.println(temp.split(":")[2]);
                    }
                           temp=br1.readLine();
             }
             }
      }
  }
  catch(Exception e)
      {
         System.out.println(e);
      }
  }

  public void display()
```

```
      }

      public void regClient()
      {
             try
             {
                    Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
                    String url="jdbc:odbc:Driver={MicroSoft Access Driver
(*.mdb)};DBQ=regclient.mdb";
                    con = DriverManager.getConnection (url, "","");
             stmt=con.createStatement();
             stmt.execute("insert into reg
values('"+username+"','"+password+"')");
                    stmt.close();
             con.close();
                    System.out.println("Client reg to server");
             }
             catch(Exception e)
             {
                    System.out.println(e);
             }
             finally
      {
       }
       }

       public  String userValidation()throws Exception
   {
       try
          {
                    Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
                    String url="jdbc:odbc:Driver={MicroSoft Access Driver
 (*.mdb)};DBQ=regclient.mdb";
                    con = DriverManager.getConnection (url, "","");
              String qs = "select  password from reg  where
 username='"+username+"'";
                    stmt = con.createStatement( );
                    rs=stmt.executeQuery(qs);
                    boolean more = rs.next();
                    String pass=rs.getString(1);
```

```
     {
            if(password.equals(pass))
            {
                        stmt.execute(qs);
                        return "AUTH";
            }
     }
   }
     catch(Exception e)
     {
     System.out.println("control comes in"+e);
     }
     finally
     {
     stmt.close();
     con.close();
     }
     return "NOTAUTH";
  }
}
```

3.3. Start

```
import javax.swing.*;
import java.awt.*;

class Start  extends JFrame
{
      Toolkit kit;
      Image im;

   Start()throws Exception
   {
       super("Third party Server");
          kit=Toolkit.getDefaultToolkit();
     im=kit.getImage("image/net.jpg");
     setIconImage(im);
     setResizable(false);
```

```
        JLabel jl=new JLabel(new ImageIcon("image/start.jpg"));
     jl.setBounds(0,0,300,300);
     getContentPane().add(jl);
        }

        public static void main(String[] a) throws Exception
        {
                new Start();
        }
}
```

Module 4: Timeserver

1. Timeserver

```
import java.util.*;
import java.net.*;
import java.io.*;
import java.text.*;

class TimeServer  extends Thread
{
        Date date;
        ServerSocket  timeServerSocket;
        Socket timeSocket;
        PrintWriter pw;
    SimpleDateFormat sdf=new SimpleDateFormat("EEE MMM d h:m:s a
 yyyy ");

        TimeServer()throws Exception
        {
                super("timeServer");
                timeServerSocket=new ServerSocket(2500);
                start();
        }

        public void run()
        {
                {
                        while(true)
                        {
                        timeSocket=timeServerSocket.accept();
                        pw=new
PrintWriter(timeSocket.getOutputStream(),true);
                        Date d2=new Date();
                                System.out.println(sdf.format(d2));
                        pw.println(sdf.format(d2));
                        }
                }
                catch(Exception e)
                {
                System.out.println("Error
!"+e.getClass()+"\n"+e.getMessage());
                }
        }

        public static void main(String[] a)throws Exception
        {
                new TimeServer();
```

Module 5: Moderator

5.1. Database

```
        import java.io.*;
 import java.sql.*;

 public class DataBase
 {
        Statement stmt=null;
    ResultSet rs=null;
        Connection con=null;
    int i=0,totalRecord;

        public DataBase()
        {
```

```
        public void   insertQestion(String qes1,int n)throws Exception
        {
                try
                {
                        Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
                        String url="jdbc:odbc:Driver={MicroSoft Access Driver
(*.mdb)};DBQ=fingerprint.mdb";
                        con = DriverManager.getConnection (url, "","");
                String qs = "select count(*) from userqes";
                        stmt = con.createStatement( );
                        rs = stmt.executeQuery(qs);
                boolean more = rs.next();
                int num=rs.getInt(1);
                qs="update userqes  set qes='"+qes1+"'  where num= "+n;
                        stmt.execute(qs);
                }
                catch(Exception e)
                {
                        System.out.println(e);
                }
                finally
        {
         stmt.close();
         con.close();
         }
         }

        public String  insertDetail(String username,String password,String
 sex,String age,String qes1,String qes2,String qes3,String qes4,String
 qes5,String ans1,String ans2,String ans3,String ans4,String ans5)throws
 Exception
        {
                try
                {
                        Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
                        String url="jdbc:odbc:Driver={MicroSoft Access Driver
(*.mdb)};DBQ=fingerprint.mdb";
                        con = DriverManager.getConnection (url, "","");
                String qs = "select * from user where
```

```
                        stmt = con.createStatement( );
                        rs = stmt.executeQuery(qs);
                boolean more = rs.next();
                System.out.println(rs.getString(1));
                        return "NOTREG";
                }
                catch(Exception e)
                {
                        System.out.println(e);
                stmt.execute("insert into user
values('"+username+"','"+password+"','"+sex+"','"+age+"','"+qes1+"','"+qes2
+"','"+qes3+"','"+qes4+"','"+qes5+"','"+ans1+"','"+ans2+"','"+ans3+"','"+ans4
+"','"+ans5+"')");
                }
                finally
        {
         stmt.close();
         con.close();
         }
                return "REG";
         }

   public String  updateDetail(String firstusername,String username,String
 password,String sex,String age,String ans1,String ans2,String ans3,String
 ans4,String ans5)throws Exception
        {
                try
                {
                        Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
                        String url="jdbc:odbc:Driver={MicroSoft Access Driver
(*.mdb)};DBQ=fingerprint.mdb";
                        con = DriverManager.getConnection (url, "","");
                String qs = "update user  set
username='"+username+"',password='"+password+"',sex='"+sex+"',age='"+a
ge+"',ans1='"+ans1+"',ans2='"+ans2+"',ans3='"+ans3+"',ans4='"+ans4+"',ans
5='"+ans5+"' where username='"+firstusername+"'";
                        stmt = con.createStatement( );
                        stmt.execute(qs);
                        return "UPDATE";
                }
```

```
          {
                System.out.println(e);
          }
          finally
        {
      stmt.close();
      con.close();
      }
          return "NOTUPDATE";
      }

      public  void  retriveDataBase()throws Exception
      {
        try
        {
                  System.out.println("control comes in method");
                  Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
                  String url="jdbc:odbc:Driver={MicroSoft Access Driver
(*.mdb)};DBQ=fingerprint.mdb";
                  con = DriverManager.getConnection (url, "","");
       String qs = "select * from user where username='saro'";
                  stmt = con.createStatement( );
                  rs = stmt.executeQuery(qs);
       boolean more = rs.next();
            for(int i=1;i<=14;i++)
            {
            System.out.println(rs.getString(i));
            }
                  stmt.close();
       con.close();
          }
         catch (java.lang.Exception e)
            {
       System.out.println("Error Inside DataBase class :\nError in
retriveDataBase\n"+e);
      }
       }

       public void deleteRecord(String username)throws Exception
       {
```

```
          {
                Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
                String url="jdbc:odbc:Driver={MicroSoft Access Driver
(*.mdb)};DBQ=fingerprint.mdb";
                con = DriverManager.getConnection (url, "","");
            String qs = "delete from  user  where
username='"+username+"'";
                stmt = con.createStatement( );
                stmt.execute(qs);
        }
        catch (java.lang.Exception e)
        {
            System.out.println("Error Inside DataBase class :\nError in
retriveDataBase\n"+e);
        }
            finally
        {
          stmt.close();
        con.close();
        }
        }

        public static void main( String argv[] )throws Exception
        {
              DataBase db=new DataBase();
              String s=
db.insertDetail("saro","saro","male","21","ds","ds","ds","ds","sa","ds","ds","
ds","ds","sa");
              //System.out.println(s);
              db.insertQestion("sasas",1);
        }
}
```

5.2. Form

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
```

```
class Form extends JDialog
{
      JLabel jl,jl2;
      JTextField jtf,jtf1;
      JButton jb;
      int n;
      String qes="",num="";

      public Form()
      {
            super(new JFrame(),"Question");
            getContentPane().setLayout(null);
            setBounds(100,100,400,150);
            jl=new JLabel("Qestion       :");
            jl.setBounds(10,40,150,30);
            getContentPane().add(jl);

            jl=new JLabel("Qestion  No  :");
            jl.setBounds(10,10,100,30);
            getContentPane().add(jl);

            jtf=new JTextField();
            jtf.setBounds(90,10,250,23);
            getContentPane().add(jtf);

      jtf1=new JTextField();
            jtf1.setBounds(90,40,250,23);
            getContentPane().add(jtf1);

            jb=new JButton("Ok");
            jb.setBounds(120,80,80,30);
            getContentPane().add(jb);

            jb.addActionListener(new ActionListener()
            {
                  public void actionPerformed(ActionEvent ae)
                  {
                        try
                        {
                              num=jtf1.getText();
```

```
                                    System.out.print(qes+"  "+num);
                                    n=Integer.parseInt(num);
                                    if(n>5||n==0)
                                    {

      JOptionPane.showMessageDialog(null,"Enter The Number b/w 1-5
");
                                          jtf1.setText("");
                                    }
                                    else
                                    {
                                          call();
                                          setVisible(false);
                                    }
                              }
                              catch(Exception e)
                              {

      JOptionPane.showMessageDialog(null,"Enter The Number ");
                                    jtf1.setText("");
                              }
                        }
                  });
                  setVisible(true);
      }

      public void call()
      {
            try
            {
                  DataBase db=new DataBase();
                  db.insertQestion(qes,n);
            }
            catch(Exception e)
            {
            }
      }

      public static void  main(String[] s)
      {
        }
}
```

5.3. Moderator

```
import java.net.*;
import java.io.*;
import javax.swing.*;
import  java.awt.event.*;

class Modurator extends Thread
{
        JFrame jf;
        JScrollPane jsp;
        JTextArea jta;
        JMenuBar jmb;
        JMenu jm;
        JMenuItem jmi1,jmi2;
        Start s1;
        ServerSocket  fingerprintserverSocket=null;
        Socket  fingerprintSocket=null;
        ModuratorServer fps;
        Start st;
        public static int totalUser=0;

        public Modurator()throws Exception
        {
                super("Modurator");
        new  ModuratorServerListen();
                st=new Start();
                jf=new JFrame("Modurator");
        s1=new Start();
        jf.getContentPane().setLayout(null);
        jf.setBounds(50,50,600,400);
        jta=new JTextArea();
        jta.append("                    MODURATOR SERVER
 ");
        jta.setEditable(false);
        jsp=new JScrollPane(jta);
        jsp.setBounds(10,10,570,320);
[illegible]
```

```
        jmb=new JMenuBar();
        jf.setJMenuBar(jmb);
        jm=new JMenu("Help");
        jmi1=new JMenuItem("Add Qestion");
        jm.add(jmi1);

              jmi1.addActionListener(new ActionListener()
          {
              public void  actionPerformed(ActionEvent ae)
              {
                      new Form();
              }
          });

              jmi2=new JMenuItem("About");
        jm.add(jmi2);

              jmi2.addActionListener(new ActionListener()
          {
              public void  actionPerformed(ActionEvent ae)
              {
                      st.setVisible(true);
              }
          });

        jmb.add(jm);
        jf.setVisible(true);
        jf.setDefaultCloseOperation(3);
        fingerprintserverSocket=new ServerSocket(1501);
              start();
        }

        public void run()
        {
              while(true)
        {
              try
              {
                      fingerprintSocket=fingerprintserverSocket.accept();
              fps= new ModuratorServer(this);
```

```
              catch(Exception e)
              {
              }
        }
        }

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

        System.out.println(InetAddress.getLocalHost().getHostAddress());
        }
        catch(Exception e)
        {
        }
        new Modurator();
        }
}
```

5.4 ModeratorService

```
import java.net.*;
import java.io.*;
import java.sql.*;
import java.util.Date;
import java.util.*;
import  java.text.*;
class ModuratorServer  extends Thread
{

        Statement stmt=null;
    ResultSet rs=null;
        Connection con=null;
    BufferedReader br=null,br1=null;;
        PrintWriter pw,pw1;
        Modurator fp;
        String msg;
```

```
        String
username="",password="",sex="",age="",ans1="",ans2="",ans3="",ans4="",
ans5="",fileName="";
        String[] qestion=new String[100];
        int qesNo=0;
        DataBase db=new DataBase();
        Random r=new Random();
        Socket  tempSock=null;
        SimpleDateFormat sdf=new SimpleDateFormat("hh:mm:ss a");

        public  ModuratorServer(Modurator fp1)throws Exception
        {
        super("Third Party Server");
        fp=fp1;
        try
                {
                        Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
                        String url="jdbc:odbc:Driver={MicroSoft Access Driver
 (*.mdb)};DBQ=fingerprint.mdb";
                        con = DriverManager.getConnection (url, "","");
                String qs = "select * from userqes";
                        stmt = con.createStatement( );
                        rs = stmt.executeQuery(qs);
                boolean more = rs.next();
                        while(more)
                {
                        qestion[qesNo++]=rs.getString(2);
                        more = rs.next();
                }
                }
                catch(Exception e)
                {
                        System.out.println(e);
                }
                finally
            {
        stmt.close();
        con.close();
        }
[illegible]
```

```
            br=new  BufferedReader(new
InputStreamReader(fp.fingerprintSocket.getInputStream()));
            start();
      }

      public void run()
      {
            String msg;
      while(true)
      {
            try
            {
                        msg=br.readLine();
                  System.out.println("Message "+msg);
            if(msg.equals("NEWUSER"))
            {
                  for(int i=1;i<=5;i++)
                  {
                        pw.println(qestion[i]);
                  }
                        pw.println("END");
            }
   /*               else if(msg.equals("spons"))
            {
                  try
                  {
                        String ip=br.readLine();
                        String port=br.readLine();
                        fp.sinfo.setSponserIp(ip);
                        fp.sinfo.setSponserPort(port);
                  }
                  catch(Exception e2)
                  {
                        System.out.println(e2);
                  }
                  System.out.println("New Sponser has been Selected ");
                  System.out.println("New Sponser Ip
:"+fp.sinfo.getSponserIp());
                  System.out.println("New Sponser port
:"+fp.sinfo.getSponserPort());
```

```
   */
                  else  if(msg.equals("NEWUSERDE"))
            {
                  username=br.readLine();
                  password=br.readLine();
                  sex=br.readLine();
                  age=br.readLine();
                  ans1=br.readLine();
                  ans2=br.readLine();
                  ans3=br.readLine();
                  ans4=br.readLine();
                  ans5= br.readLine();
                  msg=br.readLine();
                  System.out.println("Message End for NEWUSERDE
:"+msg);
                  String
token=db.insertDetail(username,password,sex,age,qestion[1],qestion[2],qest
ion[3],qestion[4],qestion[5],ans1,ans2,ans3,ans4,ans5);
                  pw.println(token);
                  System.out.println("user "+token);
                  /*if(token.equals("REG"))
                        {
                              tempSock=new Socket("localhost",1502);
                        pw1=new
PrintWriter(tempSock.getOutputStream(),true);
                              pw1.println("NEWUSERDE");
                              pw1.println(username);
                              pw1.println(password);
                              pw1.close();
                              tempSock.close();
                              }
                  */
            }
            /*else if(msg.equals("LOGIN"))
            {
                  username=br.readLine();
                  password=br.readLine();
                              fp.jta.append("\nUser Name
"+username+"\n");
                              fp.jta.append("Try to login in to Server
```

```
                              String temp=userValidation();
                              if(temp.equals("AUTH"))
                              {
                                    fp.jta.append("\nUser "+username+"
Login to Server");
                                    pw.println(temp);
                                    System.out.println("Total
Member"+Modurator.totalUser);
                                    if(Modurator.totalUser==0)
                                    {
                                          Modurator.totalUser++;
                                    pw.println(SponserInfo.SPONSER);
                                    }
                                    else
                                    {
                                          Modurator.totalUser++;

      pw.println(SponserInfo.NOTSPONSER);

      pw.println(fp.sinfo.getSponserIp());

      pw.println(fp.sinfo.getSponserPort());
                                    }
                              }
                              else
                              {
                                    fp.jta.append("\nUser "+username+"
Not Login to Server");
                                    pw.println(temp);
                              }
                  pw1.close();
                  tempSock.close();
                  System.out.println("Register to Server");
            }
      */
            else if(msg.equals("QES"))
            {
                  Random r=new Random();
                  int n=r.nextInt(4);
                              System.out.println(n);
```

```
                                    {
                                          if(n==0)
                                          {
                                                n=1;
                                          }
                                          System.out.println("control comes in
method");
                                          Class.forName
("sun.jdbc.odbc.JdbcOdbcDriver");
                                          String
url="jdbc:odbc:Driver={MicroSoft Access Driver
(*.mdb)};DBQ=fingerprint.mdb";
                                          con = DriverManager.getConnection
(url, "","");
                              String qs = "select * from user where
username='"+username+"'";
                                    stmt = con.createStatement( );
                                          rs = stmt.executeQuery(qs);
                                    boolean more=rs.next();
                                    String qes=rs.getString("qes"+n);
                                    String ans=rs.getString("ans"+n);
                                          pw.println(qes);
                                          pw.println(ans);
                                          stmt.close();
                              con.close();
                              }
                              catch (java.lang.Exception e)
                                    {
                              System.out.println("Error Inside DataBase class
:\nError in retriveDataBase\n"+e);
                              }
                  }
                        }
                        catch(Exception e)
                        {
                        }
            }
            }

      public void display()
```

```
 }
 public  String userValidation()throws Exception
   {
   try
      {
              Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
              String url="jdbc:odbc:Driver={MicroSoft Access Driver
(*.mdb)};DBQ=fingerprint.mdb";
              con = DriverManager.getConnection (url, "","");
          String qs = "select  password from user  where
username='"+username+"'";
              stmt = con.createStatement( );
              rs=stmt.executeQuery(qs);
              boolean more = rs.next();
              String pass=rs.getString(1);
      if(more)
      {
      if(password.equals(pass))
       {
        stmt.execute(qs);
        return "AUTH";
       }
      }
      }
         catch(Exception e)
       {
       System.out.println("control comes in"+e);
      }
       finally
      {
      stmt.close();
      con.close();
      }
   return "NOTAUTH";


   }
```

```
 }
```

5.5. ModeratorServerListen:

```
import java.net.*;
import java.io.*;
import java.sql.*;
import java.util.Date;
import java.util.*;
import  java.text.*;

class ModuratorServerListen extends Thread
{
      ServerSocket  moduratorServerListenServerSocket=null;
      Socket  moduratorServerListenSocket=null;
      SponserInfo sinfo=new SponserInfo();

      public ModuratorServerListen()
      {
      try
      {
            moduratorServerListenServerSocket=new ServerSocket(1502);
      }
      catch(Exception e)
      {
            System.out.println(e);
      }
            start();
      }

      public void run()
      {
            while(true)
      {
                  try
                  {

      moduratorServerListenSocket=moduratorServerListenServerSocket.a
ccept();
```

```
            new  ModuratorServerListenAction(this);
                  }
                  catch(Exception e)
                  {
                        System.out.println(e.getClass());
                        System.out.println(e.getMessage());
                  }
            }
   }
}
```

5.6. ModuratoeServerListenAction:

```
import java.net.*;
import java.io.*;
import java.sql.*;
import java.util.Date;
import java.util.*;
import  java.text.*;

public  class ModuratorServerListenAction extends Thread
{
      Statement stmt=null;
   ResultSet rs=null;
      Connection con=null;
      ModuratorServerListen msL;
   String username,password;
   ObjectInputStream objIn;
   ObjectOutputStream objOut;
   String token;

      public ModuratorServerListenAction(ModuratorServerListen msl1)
      {
            try
            {
            System.out.println("Inside Modurator Listen");
                  msL=msl1;
                  objOut=new
ObjectOutputStream(msL.moduratorServerListenSocket.getOutputStream())
;
```

```
                  objIn=new
ObjectInputStream(msL.moduratorServerListenSocket.getInputStream());
            }
            catch(Exception e)
            {
            System.out.println(e.getClass());
            System.out.println(e.getMessage());
      }
            start();
      }

      public void run()
      {
            String msg;
            try
      {
            while(true)
            {
                        msg=objIn.readObject().toString();
                  System.out.println("Message "+msg);
                  if(msg.equals("spons"))
            {
                  try
                  {
                        String ip=objIn.readObject().toString();
                        String port=objIn.readObject().toString();
                        msL.sinfo.setSponserIp(ip);
                        msL.sinfo.setSponserPort(port);
                  }
                  catch(Exception e2)
                  {
                        System.out.println(e2.getClass());
                        System.out.println(e2.getMessage());
                  }
                  System.out.println("New Sponser has been Selected ");
                  System.out.println("New Sponser Ip
:"+msL.sinfo.getSponserIp());
                  System.out.println("New Sponser port
:"+msL.sinfo.getSponserPort());
            }
```

```
		{
				try
				{
				String ip=objIn.readObject().toString();
				String port=objIn.readObject().toString();
					System.out.println("Sponser Token Send to
Member");
					System.out.println("Ip Address "+ip);
					System.out.println("Port Number "+port);
					Socket		s1=new
Socket(ip,Integer.parseInt(port));
					ObjectOutputStream	objOut1=new
ObjectOutputStream(s1.getOutputStream() );
					ObjectInputStream  objIn1=new
ObjectInputStream(s1.getInputStream());

	objOut1.writeObject(SponserInfo.SPONSER);
					objOut1.flush();
					System.out.println("Sponser Token  has
been Sent");
				}
				catch(Exception e)
				{
					System.out.println(e);
				}
		}
		else if(msg.equals("LOGIN"))
		{
			username=objIn.readObject().toString();
			password=objIn.readObject().toString();
					// msL.jta.append("\nUser Name
"+username+"\n");
					// msL.jta.append("Try to login in to Server
Time\n"+new Date());
					String temp=userValidation();
					if(temp.equals("AUTH"))
					{
						//fp.jta.append("\nUser "+username+"
Login to Server");
						//objOut.println(temp);
```

```
						objOut.flush();
						System.out.println("Total
Member"+Modurator.totalUser);
						if(Modurator.totalUser==0)
						{
							Modurator.totalUser++;

	objOut.writeObject(SponserInfo.SPONSER);
							objOut.flush();
						}
						else
						{
							Modurator.totalUser++;

	objOut.writeObject(SponserInfo.NOTSPONSER);

	objOut.writeObject(msL.sinfo.getSponserIp());

	objOut.writeObject(msL.sinfo.getSponserPort());
							objOut.flush();
						}
					}
					else
					{
						// fp.jta.append("\nUser
"+username+"  Not Login to Server");
						System.out.println("\nUser
"+username+"  Not Login to Server");
						objOut.writeObject(temp);
						objOut.flush();
					}
		}
		else  if(msg.equals("LOGOUT"))
		{
					username=objIn.readObject().toString();
					Modurator.totalUser--;
				System.out.println("Total Member Of Group
:"+Modurator.totalUser);
		}
		}
```

```
	catch(Exception e)
	{
		System.out.println("Member Has been Closed");
		System.out.println(e.getCause());
		System.out.println(e.getMessage());
	}
	}

	public  String userValidation()throws Exception
 {
	try
	{
			Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
			String url="jdbc:odbc:Driver={MicroSoft Access Driver
(*.mdb)};DBQ=fingerprint.mdb";
			con = DriverManager.getConnection (url, "","");
		String qs = "select  password from user  where
username='"+username+"'";
			stmt = con.createStatement( );
			rs=stmt.executeQuery(qs);
	 if(more)
	 {
		if(password.equals(pass))
		{
				stmt.execute(qs);
				return "AUTH";
		}
	 }
	}
	  catch(Exception e)
	 {
	 System.out.println("control comes in"+e);
	}
	 finally
	{
	 stmt.close();
	 con.close();
	 }
	 return "NOTAUTH";
   }
```

5.7. Start:

```
import javax.swing.*;
import java.awt.*;

class Start  extends JFrame
{
	Toolkit kit;
	Image im;

   Start()throws Exception
	{
	  super("Modurator");
	    kit=Toolkit.getDefaultToolkit();
	im=kit.getImage("image/net.jpg");
	setIconImage(im);
	setResizable(false);
	setBounds(100,150,550,450);
	    JLabel jl=new JLabel(new ImageIcon("image/start.jpg"));
	jl.setBounds(0,0,300,300);
	getContentPane().add(jl);
	  }

	  public static void main(String[] a) throws Exception
	  {
		new Start();
	  }
 }
```

APPENDIX 2

2. Screen shots:

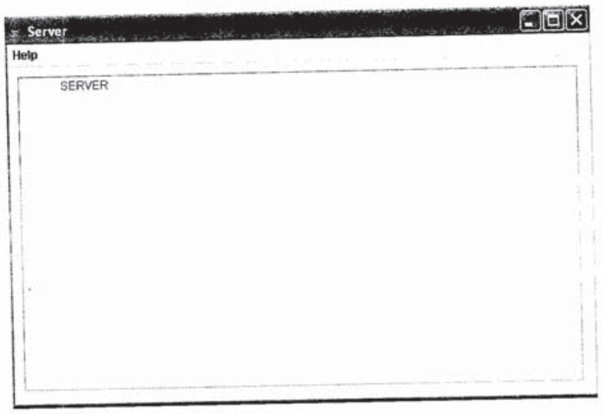


Fig A2.1 Main Server Interface

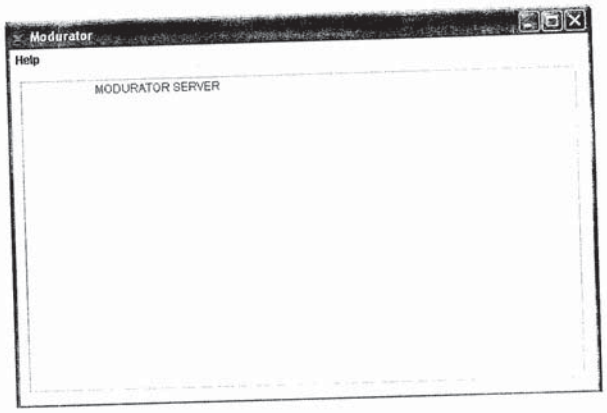
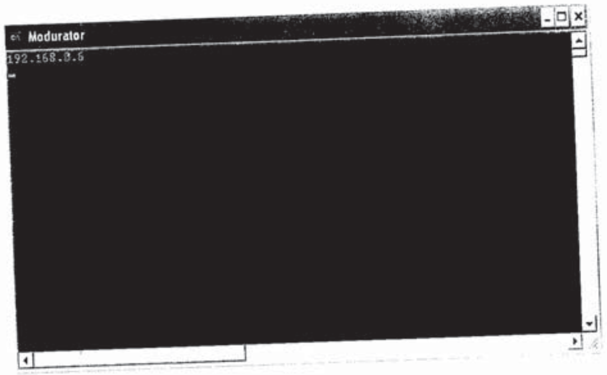
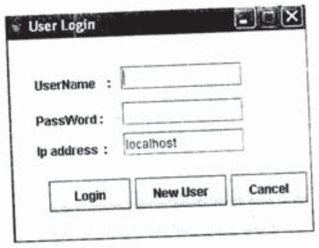


Fig A2.2 Moderator Interface



```
192.168.0.6
```

FigA 2.3 Service Details of Moderator



UserName :

PassWord :

Ip address : localhost

Fig A2.4 Client Login Window

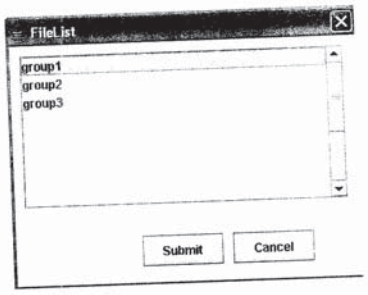
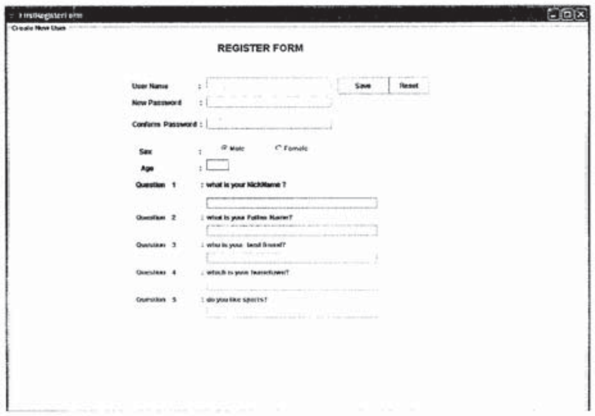
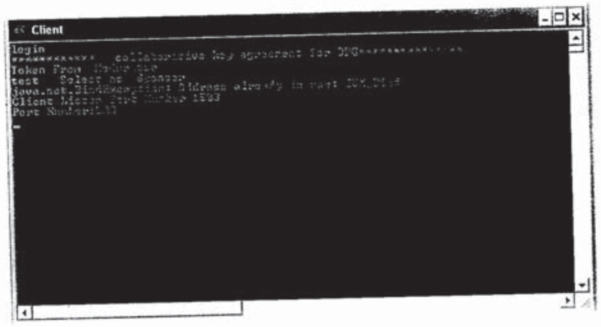


Fig A2.5 Client's Choice Window for a particular Service



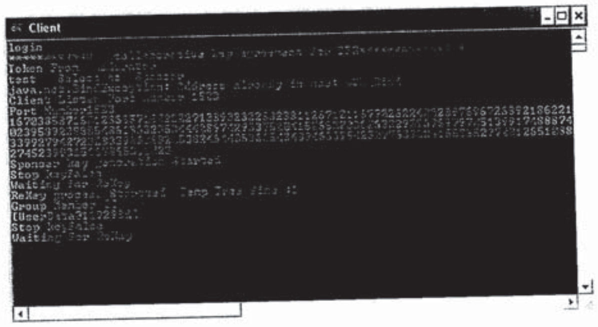
```
[illegible]
Create New User

REGISTER FORM

User Name          :              Save   Reset
New Password       :
Conform Password   :

Sex                :   ◉ Male    ○ Female
Age                :
Question  1        : what is your NickName ?
Question  2        : what is your Fathe Name?
Question  3        : who is your  best friend?
Question  4        : which is your hometown?
Question  5        : do you like sports?
```

Fig A2.6 New User Registration Window



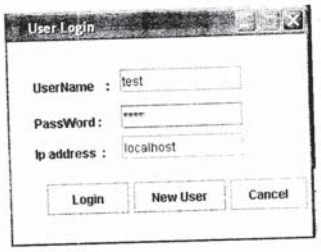
```
Client
login
*********** collaborative key agreement for DDG[illegible]
Token From [illegible]
test   Select as  Sponsor
java.net.BindException: Address already in use: [illegible]
Client Listen Port Number 1503
Port Number[illegible]
_
```

Fig A2.7 Client Behavior Window



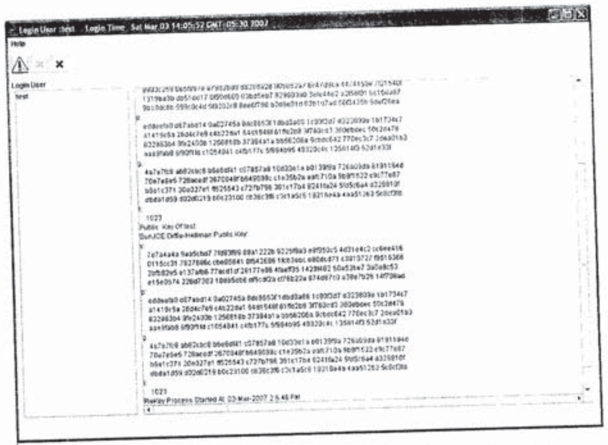
```
Client
login
[illegible] collaborative key agreement for DDG[illegible]
Token From [illegible]
test   Select as  Sponsor
java.net.BindException: Address already in use: [illegible]
Client Listen Port Number 1503
Port Number1503[illegible]2186221
167233587[illegible]188874
023959[illegible]2651838
339927962[illegible]
27452[illegible]
Sponsor Key generation Started
Stop key[illegible]
Waiting for ReKey
ReKey process Started  Temp Tree Size [illegible]
Group Member [illegible]
[UserData[illegible]
Stop key[illegible]
Waiting for ReKey
```

Fig A2.8 Client Behavior Window after Rekeying



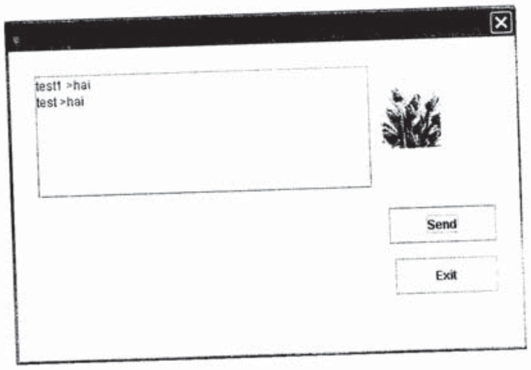
```
User Login

UserName  :  test
PassWord  :  ****
Ip address :  localhost

Login    New User    Cancel
```

Fig A2.9 Client Logs in Window



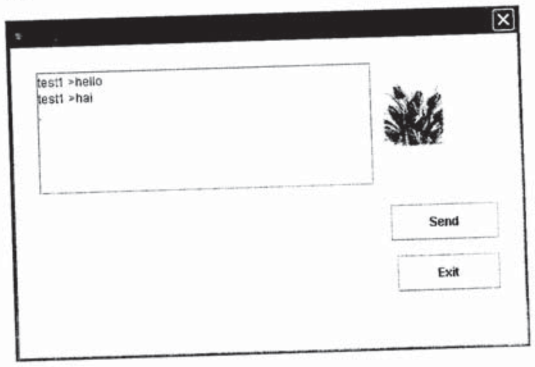
```
Login User :test   Login Time  Sat Mar 03 14:05:52 GMT+05:30 2007
Help

Login User
test
        [illegible]
        1023
        Public Key Of test
        SunJCE Diffie-Hellman Public Key
        [illegible]
        1023
        ReKey Process Started At 03-Mar-2007 2:5:46 PM
```

Fig A2.10 Client's Interface Window and the background service details by server



```
test1 >hai
test >hai

Send
Exit
```

Fig A2.11 Client1's chat window



```
test1 >hello
test1 >hai

Send
Exit
```

REFERENCES

[1] A.Niemi, J.Arkko, V.Torvinen (2002), 'Digest Authentication Using Authentication and Key Agreement'. Internet Engineering Task Force

[2] B.Tung, C.Neuman, J.Wray, A.Medvinsky, M.Hur, and J.Trostle(May 2005), 'Public Key Cryptography for Initial Authentication in Kerberos', Internet Engineering Task Force .

[3] Herbert Schildt (2002), 'The Complete Reference Java 2' (4th Ed). Tata McGraw-Hill.

[4] William H.Stallings(2000), 'Cryptography and Network Security', Pearson Education , Asia.

[5] Jonathan B.Knudsen, 'Java Network Programming' (Second Edition), O'Reilly & Associates, Inc.,

WEBSITE:

[1] www.hack.gr/users/dij/crypto/overview/diffie.html

[2] http://apocalypes.org/pub/u/seven/diffie.html

[3] www.semper.org/sierene/outsideworld/security.html