**CS 6349.001 NETWORK SECURITY**

**Project Report on**

**SECURE CHAT APPLICATION**

Submitted to

Dr. Kamil Sarac,

Associate Professor, Department of Computer Science  
Director of Education Programs, Cyber Security Research & Education Institute  
University of Texas at Dallas  
Richardson, TX 75080

By-

Sahith Katukuri (sxk145130)

Arnav Sharma (axs143930)

Table of Contents

1. Introduction ………………………………………………………………………………………………………….3
2. Project Overview and Requirements …………………………………………………………………….3
3. Project Design ………………………………………………………………………………………………………4
4. Output ………………………………………………………………………………………………………………….5

**INTRODUCTION**

In this project, we have implemented a secure chat system (say Whatsapp/Facebook or so) which will facilitate anonymous chatting amongst their buddies.

**PROJECT OVERVIEW AND REQUIREMENTS**

Main components in the system are:

1. Server
2. Client
3. All the Database/Information is stored in a text file – each line.

**Server** – We are creating a Java based server which is running on a localhost machine on port 2500.

We have multiple clients and they have their buddy lists to which they can talk and send messages without going through the server.

Our text file is being processed and being read line by line and we storing all the information in Hashmap. The text file has field for clients – username,password, userID, PortNumber.

**The system will ensure the following rules:**

1. Clients will have an account with the server to be able to use the application. It is likely that during the account opening phase, the client will obtain necessary security credentials from the server and will use them to do instant messaging with other registered users.
2. Clients will have a buddy list and will be able to do instant messaging with these people only. Users can include new registered names into the buddy list or they can remove them from their buddy lists.

**Algorithm Used:**

Symmetric Encryption : AES

Public Encryption : RSA

Message Integrity : HashFunction with SHA-256

**Methodology:**

We are writting the code in Java and use the AES Library to encrypt messages.

Server will act as KDC

KKDC – Public key of the Server

KA – Master Key of Client (Hash of Password of ClientA)

KB – Master Key of Client (Hash of Password of ClientB)

SA – Temporary Secret Key

KAB – Session Key for communication between ClientA and ClientB

TGT = KKDC {ClientA , SA}

* The username and passwords of all the clients are stored in a file inside the server and is accessible by the server only.
* We will be using a strong client password which will be hashed and made to a 128-bit password.
* The server will store the hashed password value in the database.
* The public key of the server will be known to all the clients.
* Hence, clients will be using the servers public key (128-bit / 16 bytes long) to encrypt username, password, ” I am ClientA ”, IP Address and Port Number using the AES encryption.
* Server will decrypt using its own private key and will read the username and password and match it against the username, password which is stored in the server.
* Now the server will generate a Temporary Secret Key (SA) and TGT (Ticket granting Ticket) and encrypt it using KA and send it to ClientA.
* The ClientA will decrypt it and send TGT, ClientB (It wants to talk to ClientB) as clear text and Time Stamp encrypted by SA.
* The server now will encrypt ClientB’s IP Address, Port Number, KAB (It generated a Session Key), KB{“ClientA”, KAB}
* ClientA then decrypts message using SA.
* Now there will be communication between ClientA and ClientB.
* Now ClientA will send KB{“ClientA”, KAB} and KAB {Time Stamp} to ClientB
* Client would Authenticate itself by KAB {Time Stamp+1}
* Now that the trust is established between both Clients, there can secure communication between both Clients and the messages are secured using the session key KAB.
* Here we are using the **Kerberos** Protocol for Secure Communication.

**Requirements:**

**Authenticaton:** As specified authentication is performed using challenge

messages encrypted with public keys.

**Message Integrity:** Hmac with sha-256 using shared keys are used for obtaining

message digests.

**Encryption:** Every message is either encrypted with symmetric encryption or

asymmetric encryption. Symmetric encryption is done using Secretkeys. Asymmetric

encryption is done using public keys.

**Design:**

**Thread management:**

1. Client initiates the request. A while loop will be running in client prompting the user to search for an item in server. The item name and server should be given by user.
2. In server for every request, a new thread is spawned. This thread renders the requests from client and sends products.
3. Technically, for every request, a socket connection is opened and a separate thread handles the messages through this socket.
4. One thread per request design simplifies management of socket sessions and key exchange.

**Socket connections:**

1. TCP sockets are used for communication
2. For every request, a socket connection is opened and a separate thread handles the messages through this socket.

C:\Users\Arnav\Desktop\Untitled Diagram1.jpg

**Attack Prevention:**

* Trudy will not be able eavesdrop the communication because during the authentication it is encrypted by server’s public key which can only be decrypted by private key of server.
* The response message is also encrypted by the master key of the ClientA which is hard to guess.
* The Integrity of the message is secured by encrypting using DES CBC mode and CBC residue.

**How to run:**

Compile all java files = javac \*.class

To run srver = java MainServer

To run clients = java Client (clientid char ‘A’ – ‘J’) (port 6553-6562 resp)