SECURE INSTANT MESSAGING SYSTEM

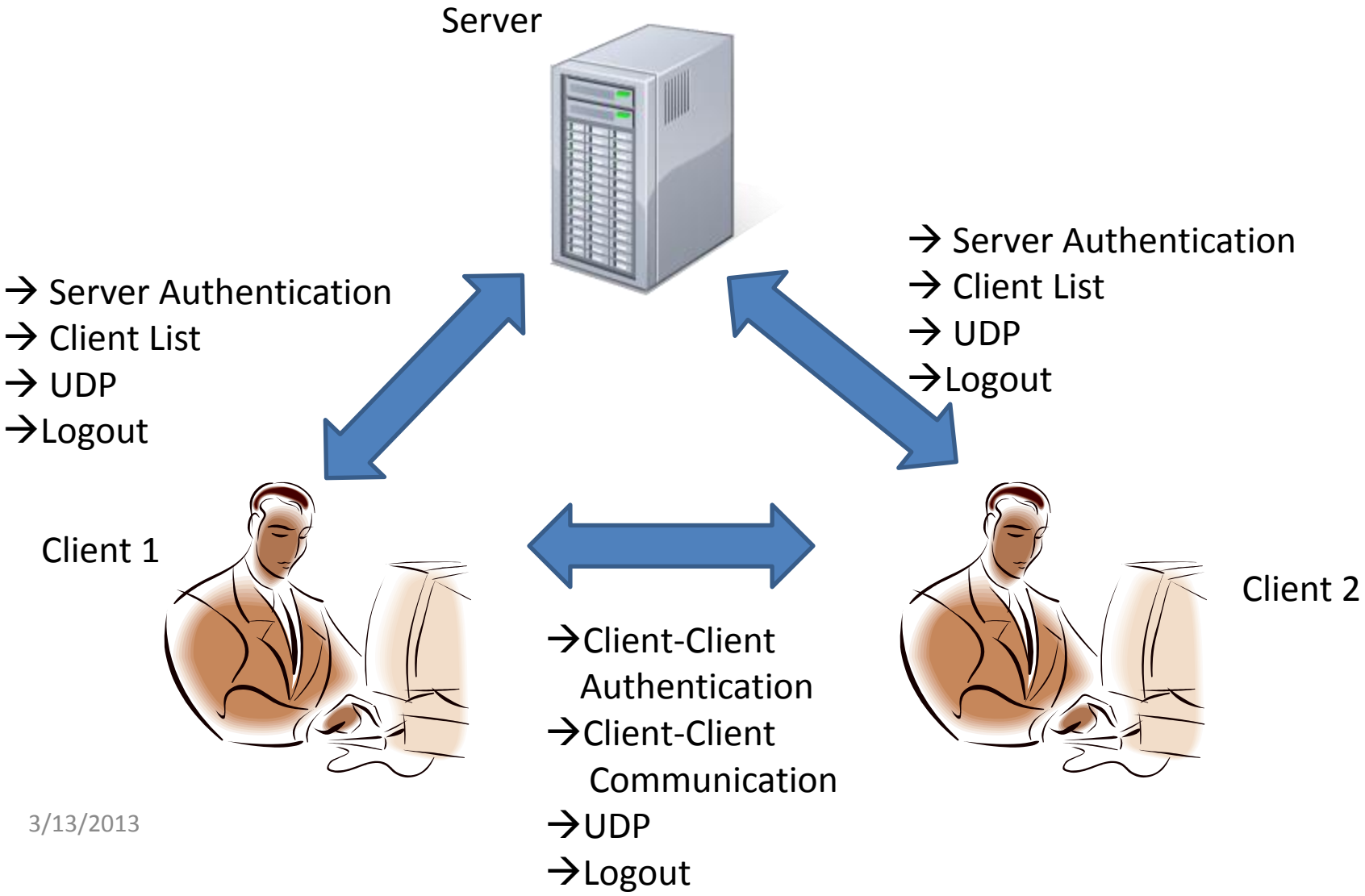
PS4 – Final Report
CS6740 Network Security
Arpit Mehta
Ananthananrayanan Balachandran
Team 5

ARCHITECTURE OVERVIEW

- Server:
  - Stores the weak password hash (W) of the registered clients
  - Stores the hash of client password and username (W')
  - Facilitates client authentication and client-to-client shared key distribution
  - Provides each client with current list of online clients on request
  - Not involved in forwarding messages.

- Client:
  - Multiple instances
  - Each authenticates itself to the server
  - Requests the server for the list of all registered clients
  - Client-To-Client Communication.
  - Every client knows the other client with which it want to chat

ARCHITECTURE OVERVIEW



ASSUMPTIONS & SOME IMPLEMENTATION DETAILS

- A user only needs to remember his/her username and password
- The client application doesn't store the client password
- The client application doesn't need to know it's private key but it knows the server public key
- Encryption and Decryption using symmetric keys is done using 128 bit AES keys using AES/CBC/PKCS5Padding mode
- The Server creates a database of the registered clients, when the SIMHelper.java application is run. This application does the following
  - Creates the server RSA public and private key files
  - Creates RSA public and private key files for each user
  - Creates PBE file for each client which has W' = hash(username | password)
  - Creates a ClientNameList for the server which contains the following information for each client:
    - Username
    - Username length
    - Password Hash [W = Hash(password)]
    - Password Hash (W) Length

NOTATIONS

A: Client A, has secret password (pwd)

S: Server

g, p: Shared values, used to establish session keys

a, s: are secret Diffie-Hellman numbers chosen by A & S respectively

W: Secret derived from Hash(pwd)

W': Secret derived from Hash(username | pwd)

Server also has a preconfigured map of W & W' of each client

C_1, C_2: Challenges between Clients and Server

Y: W'{Ka}, i.e. Client's private key encrypted with W'

CLIENT-SERVER AUTHENTICATION

- Two way Authentication using Augmented Strong Password Protocol
- Steps:

  A → S: Login

  S → A: cookie = hash(client IP, client port, server secret)

  A → S: cookie, {"username", C_1}Ks, W{g^a mod p}

  S → A: W{g^b mod p}, (g^{as} mod p){Y, C_1 – 1}, C_2

  Established Session Key, $K_{AS} = g^{as}$ mod p

  A → S: [hash((g^{as} mod p)]Ka, C_2 – 1

Client is now authenticated at the server.

PROTOCOL PROPERTIES

- Mutual authentication between client and server
- Defense against DoS attack by the use of stateless cookies
- Protection against weak password by using augmented strong password protocol
- Endpoint Hiding by encrypting the usernames during client-to-server authentication
- Perfect Forward Secrecy by forgetting the Diffie-Hellman key parameters once the authentication is done
- Secure against server break-in. An attacker cannot learn the password from the server information and, hence, cannot impersonate another user

POST AUTHENTICATION – LIST UPDATE

Once the client-to-server authentication is done, the user can request the server for the current list of online users by issuing the “list” command on the client application console.

Steps:

A → S: $K_{AS}\{list\}$

S → A: $K_{AS}\{list\ of\ online\ clients\}$

N is a Nonce to ensure the freshness of the message and to prevent replay attack

CLIENT-TO-CLIENT AUTHENTICATION

- If client “A” wants to chat with an online client “B”, “A” first sends a request message to the server that it wants to chat with “B”
- The server then issues a ticket to A to chat with B.
  ticket2B = {Kab, “A’s username”}Kb
- Stateless cookies are used in order to prevent DoS attacks
- The cookie is a function of the IP address and a secret known only to B.

Notations:

K_A : A’s public key

K_B : B’s public key

K_S : Server’s public key

c: cookie = hash(IP address, secret)

CLIENT-TO-CLIENT AUTHENTICATION

- Steps:

  Chat Request to Server:

  - A → S: {N1, "A wants to talk to B"}Ks
  - S → A: {N1, "B", K_{AB}, ticket2B} K_A

    where ticket2B = {K_{AB}, "A"} K_B

  Client-To-Client Mutual Authentication:

  - A → B: ticket2B, K_{AB}{N2}
  - B → A: K_{AB}{N2-1, N3}
  - A → B: K_{AB}{N3-1}

  Encryption using Kab

CLIENT-TO-CLIENT SESSION KEY ESTABLISHMENT

- When “A” wants to chat with “B, it establishes a Diffie-Hellman session key with “B” using the shared key provided by the server.
- Steps:

  A → B: $K_{AB}\{\text{“A”}, g^a \bmod p\}$

  B → A: $K_{AB}\{\text{“A”}, g^b \bmod p\}$

  A → B: $\text{Hash}\{g^{ab} \bmod p, N\}$

  B → A: $\text{Hash}\{g^{ab} \bmod p, N+1\}$

  Established Session Key for further communication is:

  Ksession: $g^{ab} \bmod p$

CLIENT-TO-CLIENT CHAT SESSION

Messages between clients are encrypted using Diffie-Hellman shared session key established between A and B.

A → B: $K_{session}$\{plaint text, timestamp\}

B → A: $K_{session}$\{plaint text, timestamp\}

Where $K_{AB} = g^{ab} \bmod p$ is the shared key.

CLIENT-SERVER LOG OFF

- Client needs to initiate the log off process from the server by sending the “logout” message
- Steps:

  A → S: K_{AS}{A, “logout”}

K_{AS}: Session key established between A and S

PROTECTION AGAINST ATTACKS

- DoS
  - Server denies client log in after 3 attempts for time t
  - Server checks for half open connections regularly and terminates them if found.
  - Stateless cookies are used by a client to prevent certain narrow set of DoS attacks

- Perfect Forward Secrecy

  Each client forgets g^{ab} mod p and their private a and b respectively at the end of a session, so that, there is no way for anyone to reconstruct g^{ab} mod p from knowledge of both long term private keys and recorded encrypted conversations

PROTECTION AGAINST ATTACKS

- Trusted Server

  If Alice wants to talk to Bob, she will send a request to the Server, showing she wants to talk to Bob. The server issues a “ticket2B” and a shared key .Alice and Bob will then use this shared key to establish a secure session to talk to each other, and these messages are not routed through the server. So the Server has no opportunity to decrypt the communication between Alice and Bob.

- End-Points Identity Hiding:

  In both Client-To-Server and Client-To-client communication, the identities are not transmitted over the clear. The usernames are always encrypted and an attacker will not be able to discover the identities of them unless he can obtain the corresponding key and decrypt them.

PROTECTION AGAINST ATTACKS

- Offline attacks on server

  For offline password guessing, since our protocol is based on strong password protocol, which is secure against offline password guessing. Particularly, they use Diffie-Hellman to establish the session key and the Diffie-Hellman values are encrypted with W.