

# *Architecture and Design for a Secure IM*

## *-Team 04: Discovery*



*Carl Nayak,*

*Swanand Abhyankar*

*Snowvee Gonsalves*

*Sonali Kudtarkar*

# Features

- Secure Communication :  
Authentication, Confidentiality and Integrity are maintained.
- “Strong” password protocol :  
Protection even if a relatively weak password is used using salt and UNIX server approach.
- End-to-end security :  
Securing messages cryptographically across all points between an originating user and the intended recipient.
- Perfect Forward Secrecy :  
The property that the compromise of long-term keys does not compromise previously established session keys is achieved using Diffie-Hellman approach and forgetting session keys after session is over.
- DoS Attack protection :  
Provides limited protection from DoS attacks via the use of a form of SYN cookies, using MAC to prevent tampering of messages being sent and checking the validity of messages/commands for a given phase of a client.
- Identity Hiding :  
Assurance that identities of the communicating parties are only known to server and the communicating parties.
- Commands for each phase :  
Request for logged-in user list, client logout etc. is done via commands exchanged between client and server. The validity of a command is checked for a client in a particular phase. This will help in preventing DoS attacks, caused by issue of invalid commands for a given client phase.

# Phases of the Protocol

- Phase 1 : Authentication between Client-Server
- Phase 2 : Session Key Establishment and request for logged-in user list.
- Phase 3 : Public Key Distribution for Client-Client communication
- Phase 4 : Client-Client Authentication and subsequent communication

# Assumptions of the protocol

- Public key of the server is well known to all clients.
- Users are already registered with the server. New user registration is not supported. Registration process is done offline.
- Keep alive message interval is set to 2 minutes.
- Client workstation and its applications can be trusted.
- Server and its applications can be trusted.

# Notations

- $A, B$  : **Two IM users (Alice and Bob respectively).**
- $S$  : **IM Server.**
- $N_a, N_b, N_s$  : **Nonce generated by A, B and Server respectively.**  
(this is done using SecureRandom class in java.security library for all random challenge/nonce/random number generation.)
- $ID_a, ID_b$  : **User ID of A and B respectively (unique within the IM service domain).**
- $K_{a+}$  : **public key of A.**
- $K_{a-}$  : **private key of A.**
- $K_{as}\{data\}$  : **Symmetric (secret-key) encryption of data using key shared between Server S and client A.**
- $\{data\}K_{a+}$  : **Asymmetric (public-key) encryption of data using A's public key  $K_{a+}$ .**
- $\{data\}K_{s+}$  : **Asymmetric (public-key) encryption of data using S's public key  $K_{s+}$ .**
- $[X]SHA1$  : **MAC output of data X using SHA1.**
- $K_{s\_as}$  : **shared session key between A and S**

## Phase 1 : Authentication between client - server

- Step 1 :  $A \rightarrow S : \{K_{as}\}K_{s+}, K_{as}\{ID_a, K_{a+}, \text{password}, N_a\}$
  - Step 2 :  $A \leftarrow S : \{N_s\}K_{a+}, K_{as}\{N_a + 1\}$
  - Step 3 :  $A \rightarrow S : K_{as}\{N_s + 1\}$
- ❑ Challenge/nonce usage for prevention of replay attacks.
  - ❑ Shared session key only known to server and client.
  - ❑ No one can see the content of the messages encrypted using shared session key and replay attack is prevented.

# Implementing cookies to prevent DoS attacks

- If time permits : We are using the stateless Cookie sent by server which will include the client A's IP address, and then client has to come back with the same cookie again to authenticate himself. It prevents attack of spoofing A's address by intruder T

Cookies are implemented as follows to prevent DoS attack:

A --> S : {Kas}Ks+, Kas{IDa, Ka+, password, Na}

A <-- S : Stateless Cookie

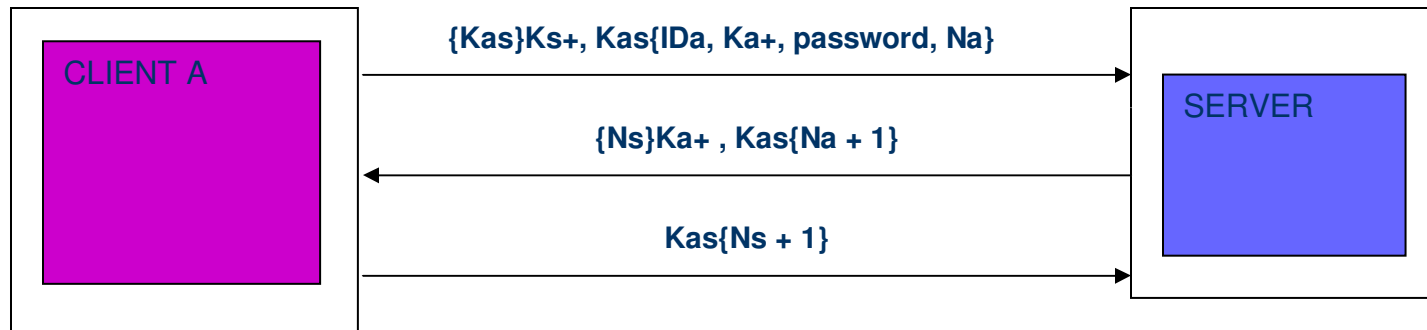
step 1 : A --> S : {Kas}Ks+, Kas{IDa, Ka+, password, Na}, Cookie

.

. (as previous slide)

.

# Client-Server Authentication



## Phase 2:

### Session Key establishment and request for logged-in user list.

For session key between Server and Client we will reuse  $K_{as}$  established from phase one without compromising perfect forward secrecy as this session key will always be different for each new session and the session key is forgotten after the current session.

Step 4 : A  $\leftarrow$  S :

$K_{as}\{\text{List of users currently logged in}\},$   
 $[\text{List of users currently logged in}]_{\text{SHA1}}$

Step 5 : A  $\rightarrow$  S :

$K_{as}\{\text{A wants to talk to B}\},$   
 $[\text{A wants to talk to B}]_{\text{SHA1}}$

- ❑ Messages are encrypted using shared session key between client-server.
- ❑ Message integrity is maintained via the use of SHA1 for MAC.
- ❑ Before step 4, a command will be issued by the client requesting for logged-in user list, logout etc.

# Session Key establishment between Client – Server and List Request



## Phase 3 : Public Key Distribution between the clients

Step 6: B  $\leftarrow$  S :

$K_{bs}\{K_{a+}, ID_a, A's\ NetAddr\},$   
 $[K_{a+}, ID_a, A's\ NetAddr]SHA1$

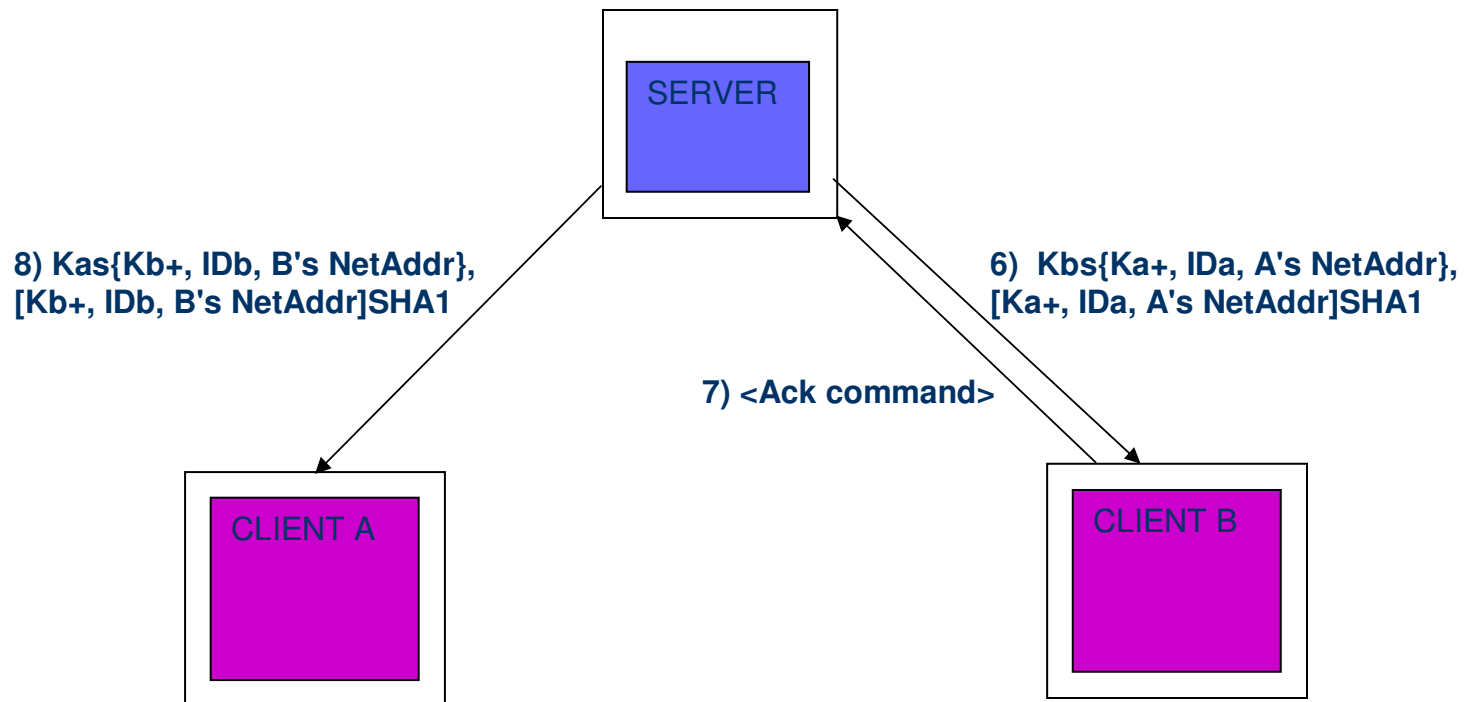
Step 7: S  $\leftarrow$  B : <send acknowledgement command>

Step 8: A  $\leftarrow$  S :

$K_{as}\{K_{b+}, ID_b, B's\ NetAddr\},$   
 $[K_{b+}, ID_b, B's\ NetAddr]SHA1$

- ❑ Messages are encrypted using shared session key between client-server without compromise of perfect forward secrecy.
- ❑ Message integrity is maintained via the use of SHA1 for MAC.

# Public Key Distribution between the clients



## Phase 4: Client-Client Authentication

Step 9 : A --> B : {Kab}Kb+, Kab{Na}

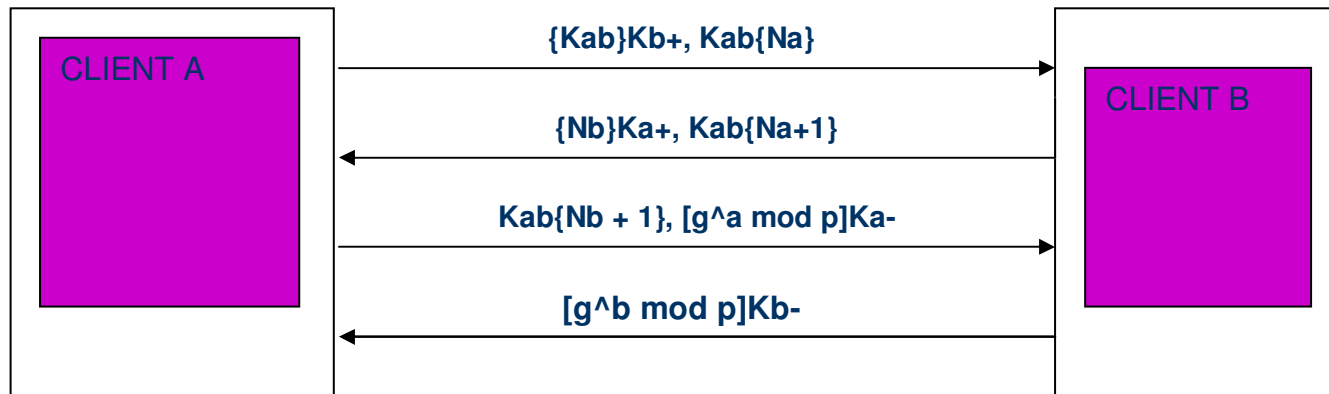
Step 10 : A <-- B : {Nb}Ka+, Kab{Na+1}

Step 11 : A --> B : Kab{Nb + 1}, [g^a mod p]Ka-

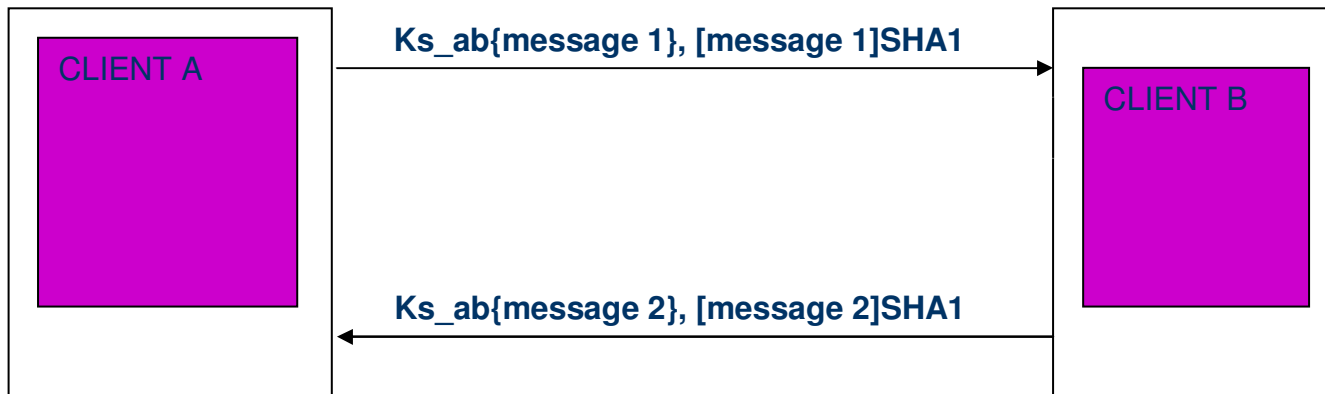
Step 12 : A <-- B : [g^b mod p]Kb-

- ❑ Challenge/nonce usage for prevention of replay attacks.
- ❑ Shared session key **Kab** only known to A and B but this was negotiated by A alone. Hence we use Diffe-Hellman in order for B to contribute towards the establishment of shared session key. Thus maintaining perfect forward secrecy by forgetting constants a, b after the shared-key has been established.
- ❑ No one can see the content of the messages encrypted using shared Diffe-Hellman key.
- ❑ The session key established using Diffe-Hellman is denoted by Ks\_ab.

# Client-Client Authentication



# Client-Client Communication



- ❑ The messages that are exchanged are encrypted using session key  $Ks_{ab}$ .
- ❑ Message integrity is maintained via the use of SHA1 for MAC.
- ❑ Identity Hiding, End-to-End security, Secure Communication and Perfect Forward Secrecy is established using this protocol.

# Known Flaws

- DoS attacks
  - Blocking server ports has limited protection
  - Sends bogus/replayed messages to server will cause the server to waste time with decryption of these messages (if cookies are not implemented)
- Weak password protection is implemented in a limited manner.
- Online password attack protection not implemented