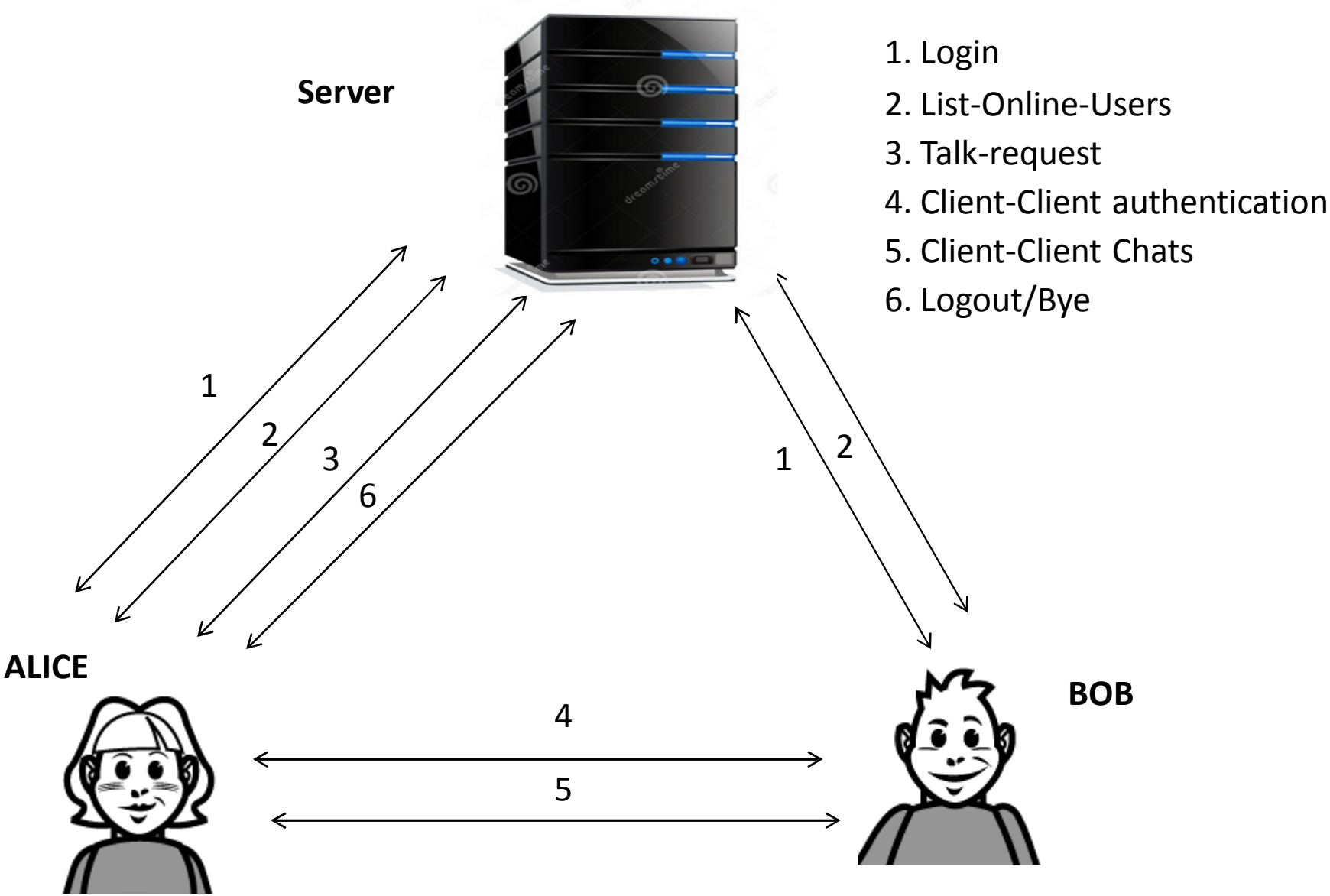
SECURE INSTANT MESSENGER

- **Bhavik Gandhi**
- **Mirav Gokani**

Architecture



Protocols

- Login (client - server)
- List-Online-Users (client - server)
- Talk Request Protocol(client – server)
- Client - Client Authetication(client- client)
- Client – Client Communication (client - client).
- Logout (client - server)

Message Types

- LOGIN
- LIST
- TALK-REQUEST
- SEND USER <message>
- LOGOUT/BYE

Challenges

- Mutual authentication (client - server)
- Mutual authentication (client - client)
- Message Integrity
- Confidentiality
- Session key establishment for client-server & client-client communication
- Perfect forward secrecy
- Prevention against weak passwords
- Protection against Denial Of Service attacks

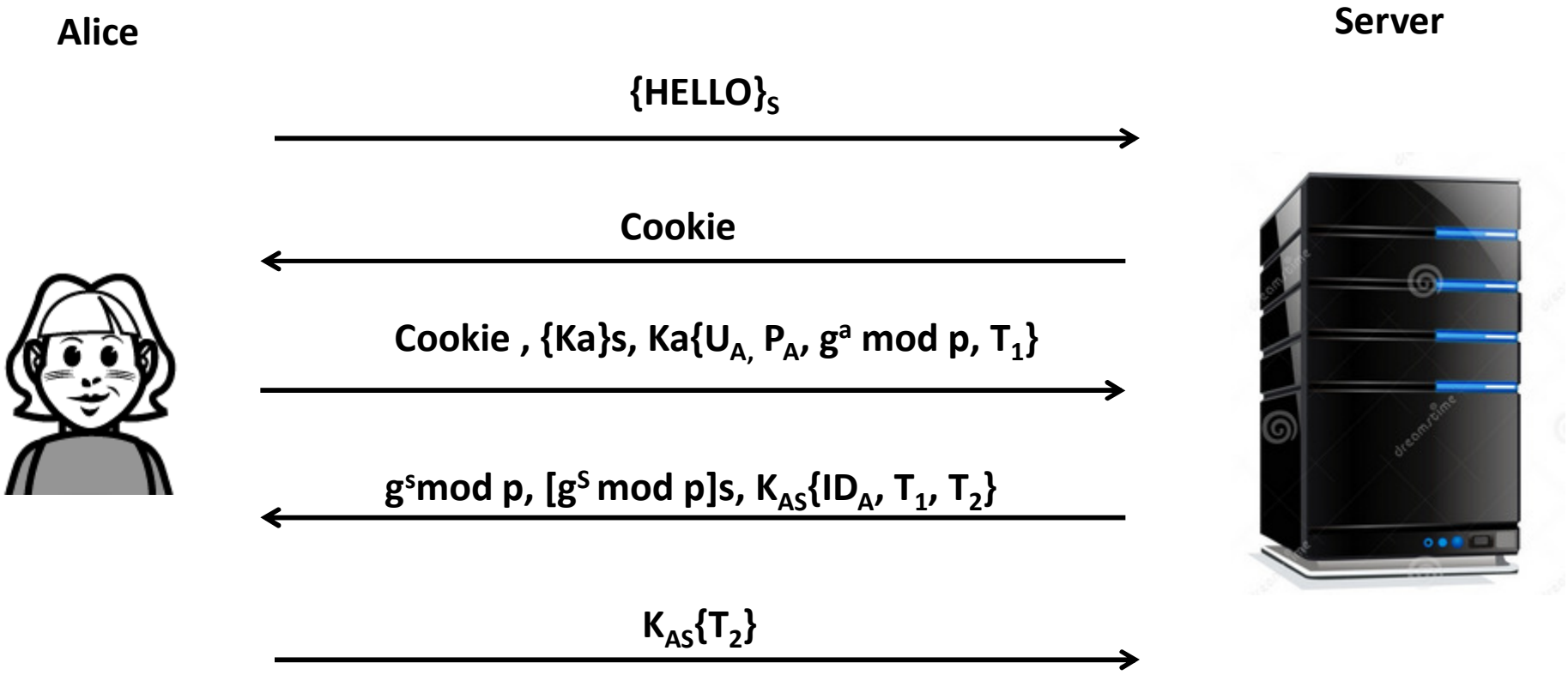
Assumptions

- Users are pre-registered
- User only need to remember password and username
- Server stores list of registered users along with their password hash
- Client knows only public key of server

TERMINOLOGY

- A,B,S – Two Clients (Alice and Bob) and Server
- ID_A , ID_B – An unique user ID of A and B
- K_{AS} – Symmetric Key between A & S
- K_{AB} – Symmetric Key between A & B
- K_{BS} – Symmetric Key between B & S
- K{M} – Symmetric Key encryption of Message M using Share secret K
- $\{M\}_A$ – Asymmetric encryption of message M using the public key of A
- S_{AB} – Temp. Session key shared by A and B, generated by S.
- T_n – Timestamps using epoch time

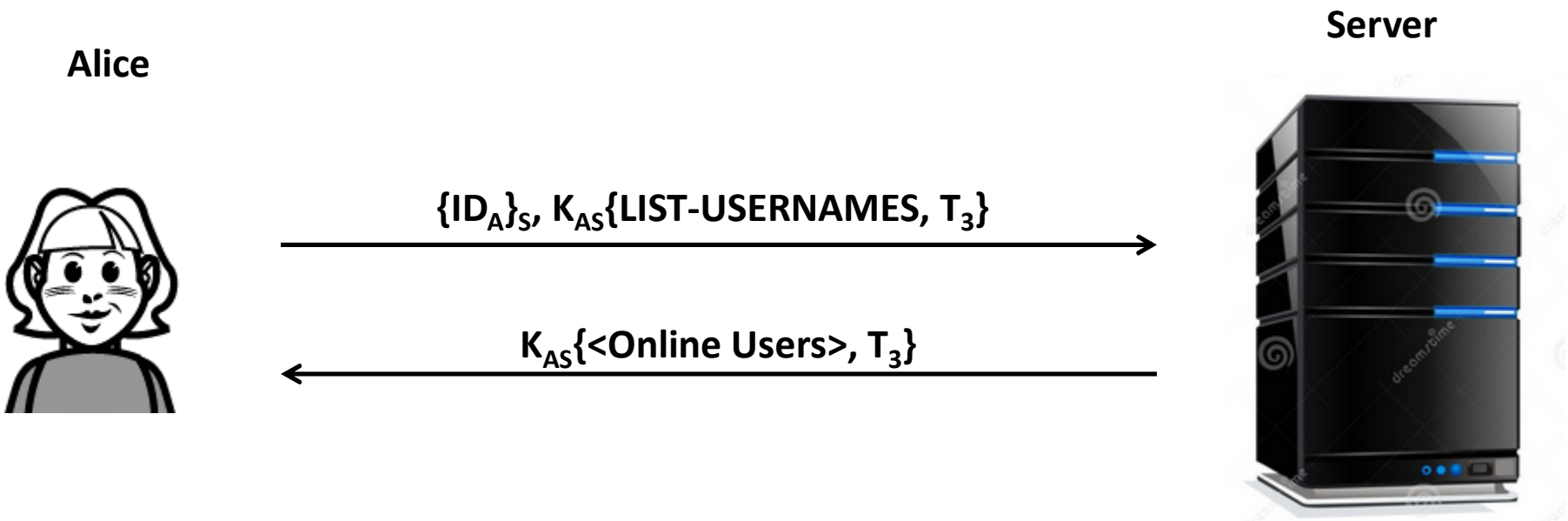
Login



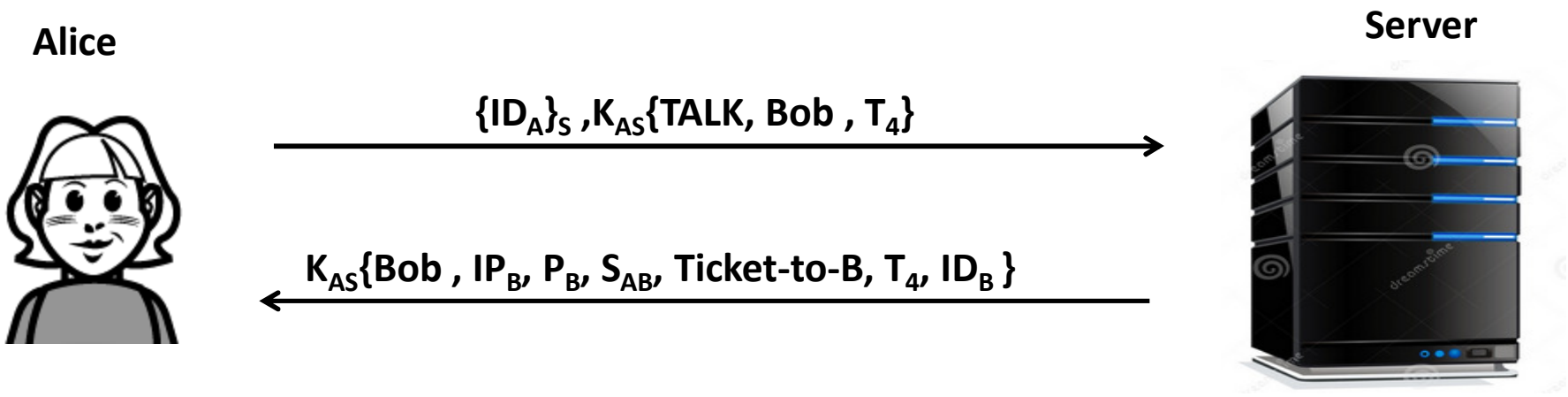
Cookie = Hash{IP_A, $Port_A$}
P_A = Hash{$Password_A$}

{HELLO}$_S$ = Login request sent by Alice
$K_{AS} = g^{as} \bmod p$
Ka = Secret Symmetric generated by A

List-Online-Users



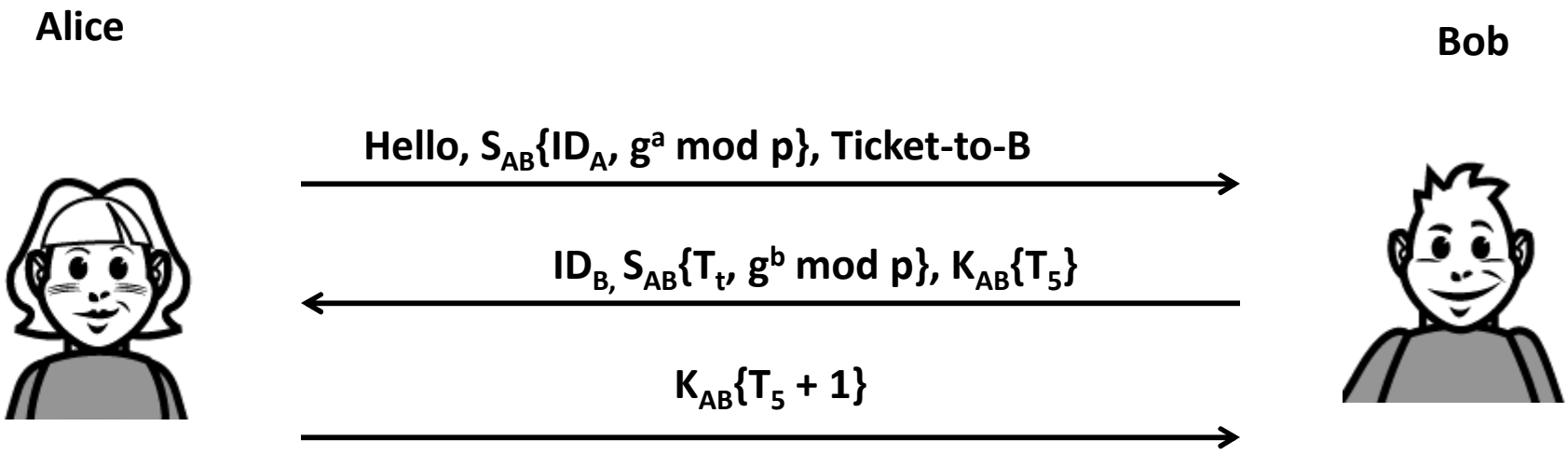
Talk Request Protocol



$$Ticket\text{-}to\text{-}B = K_{BS}\{Bob, Alice, ID_A, S_{AB}, T_t\}$$

T_t = Timestamp of ticket generation

Client Client Authentication Protocol



Peer to Peer Communication

ID not required since we are using TCP connection

Alice

Bob

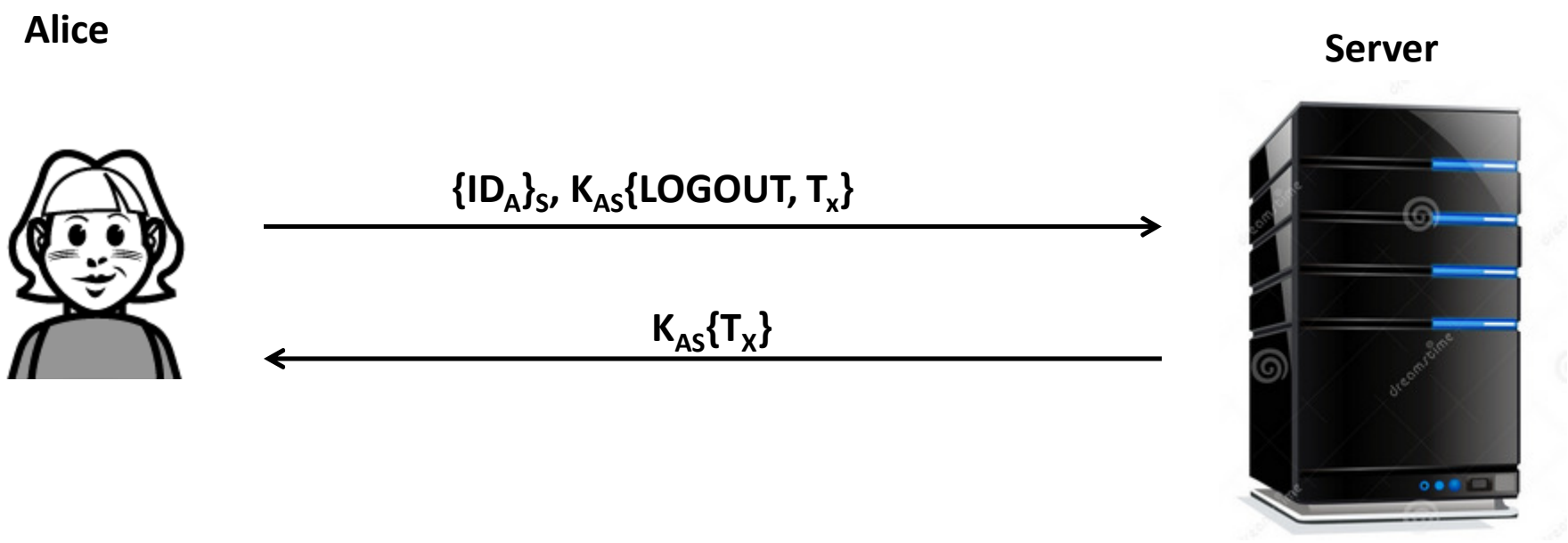
$ID_{A,}\ K_{AB}\{<Message>, HMAC(K_{MAC,}\ message), Tx\}$

$ID_x, K_{AB}\{BYE, <Message>, HMAC(K_{MAC,}\ message), Tx\}$

$K_{MAC} = \sout{f(K_{AB})} = K_{AB}$ $\qquad K_{AB} = g^{ab} \bmod p$

ID_X = ID of A or B, sending BYE message to end communication between themselves

Logout



DESTROY KEYS AFTER LOGOUT / BYE

Assuming Alice logs out or Says BYE to Bob

- K_{AS} – Symmetric key between Alice and Server on Logout
- K_{AB} – Symmetric key between Alice and Bob from both Alice and Bob on BYE message
- ID_A – Alice's unique ID given by Server

Cryptographic Algorithms

- RSA – Initial secret Key establishment
- Diffie Hellman – Session Key Exchange (p, a and b large enough to be calculated by attacker)
- AES with CBC mode – 256 bits AES Encryption
- CBC Mode – Block size 64bits
- HMAC –SHA1 – Message Integrity
- MD5 hash – For storing passwords on Server

Security

- Mutual Authentication - Client-Client & Client-Server, Session key established between parties
- Message Integrity – Used MAC, MAC key generated using shared secret key
- Replay Attacks – Used Timestamps to ensure freshness
- Reflection Attacks – Initiator is authenticated first to receiver
- Confidentiality – Messages are encrypted using AES with CBC, Key established by Diffie Hellman key-exchange scheme
- Resistant to DOS Attacks – Delayed authentication & Cookies
- Perfect Forward Secrecy – Destroy DH parameters & Session Keys
- Identity Hiding - User ID and Passwords are encrypted

ISSUE -1: Protection against weak passwords

- Password Hashes are stored at server and not plaintext passwords
- Users sends the Hash of passwords
- Client’s Hashed passwords are encrypted with the server’s public key
- Only Server can decrypt the Hash

ISSUE -2: Resilient to DOS Attack

- Delayed authentication
- Used various levels for authentication
- SYN flooding - Prevented by using cookies
- Smurf attack prevention – No broadcasting of messages, Clients and Server can use Firewall
- Firewall not implemented in design

ISSUE – 3: End point hiding

- Usernames are never sent in plaintexts
- Usernames sent are always encrypted
- Clients are given unique IDs every time they are authenticated with server
- We can use IPSEC, but not implemented in our design

ISSUE – 4: Server/Workstation trusted/not trusted by users

- If trusted: Communication is encrypted using the shared secret keys
- Server cannot decrypt communication between clients: New secret keys are established between clients to exchange messages
- If workstations compromised: Secret keys and their parameters are destroyed
- Passwords or usernames are not stored on client's workstation

Design Flaw -Login

Alice

Server

$\{HELLO\}_S$

Cookie

Cookie , $\{Ka\}s$, $Ka\{U_A, P_A, g^a \bmod p, T_1\}$

$g^s \bmod p$, $[g^S \bmod p]s$, $K_{AS}\{ID_A, T_1, T_2\}$

$K_{AS}\{T_2\}$

Replay K_{AS} $\{T_2\}$ and the attacker will see logged in message but no attacks are possible

$Cookie = Hash\{IP_A, Port_A\}$

$P_A = Hash\{Password_A\}$

$\{HELLO\}_S$ = Login request sent by Alice

$K_{AS} = g^{as} \bmod p$

Ka = Secret Symmetric generated by A

Questions?

THANK YOU ☺