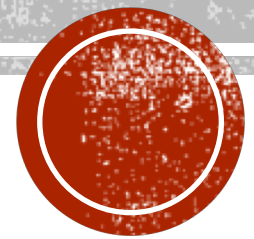
FINAL PROJECT

Yunfan Tian & Xiang Zhang



ARCHITECTURE

- Server:
- Authenticate with Client
- Save User name and password hash in database
- List of User name
- Sign up for new users
- Save list of peer-to-peer connections

ARCHITECTURE

- Client:
- Authenticate with Server
- Sign up
- Log in
- List
- Send message
- Log out

TYPES OF KEY

- Server - Client
- Asymmetric keys (RSA)
  - Server has private key and Client has public key
- Client – Client
  - Exchange symmetric key through server (key establish)
- Session keys
  - Diffie-hellman generate session key for each session

ASSUMPTIONS

- Assume attackers cannot change the source code of server and client
- Assume Server has private key and the key is absolutely safe.
- Assume Server has user-password database, attacker can't modify database
- Assume client has server's public key, attackers cannot change it
- Assume attackers have limited resource to crack password
- Assume attackers only attack on network, no side channel attack or system attack

PROTOCOL USED

- RSA signature and verification for mutual authentication
- Diffie- hellman key exchange (session key establish protocol)
- Needham–Schroeder protocol (authentication protocol)
- Bcrypt: to strength weak password
- AES (message encrypt and decrypt)

NOTATION AND LOGIN

- W is password for client, S is server RSA key
- Server stores RSA private key, User-hash(W) database
- c_1 and c_2 are challenges of client and server
- K_{AB} is shared key generated by $K_{AB} = K_A \oplus K_B$
  - Every time you login or sign up, you get different K_A and K_{AB}

Login and sign up:

$$A \rightarrow S : \{A, K_A, W_A, c_1\}_S$$

$$A \leftarrow S : K_A\{A, c_1, c_2\}, [K_A\{A, c_1, c_2\}]_S$$

$$A \rightarrow S : \{A, c_2\}_S$$

CLIENT TO CLIENT CONNECTION

- Expanded Needham-Schroeder

$A \rightarrow S : \{A, B\}_S$

$A \leftarrow S : K_A\{B\ identity, K_B\{A\}\}$

$A \rightarrow B : K_B\{A\}$

$A \leftarrow B : K_B\{A, N_B^{'}\}$

$A \rightarrow S : \{A, B, N_1, K_B\{A, N_B^{'}\}\}_S$

$A \leftarrow S : K_A\{N_1, B, K_{AB}, TicketToB\};\ TicketToB = K_B\{K_{AB}, A, N_B^{'}\}$

(B must verify $N_B^{'}$ in TicketToB, prevent T steals A's key and impersonate A even after A changes it's key)

$A \rightarrow B : TicketToB$

$A \leftarrow B : K_{AB}\{N_2\}$

$A \rightarrow B : K_{AB}\{N_2 - 1\}$

SESSION KEY AND LIST

- Diffie-Hellman key exchange, verify the timestamp for every message

$$A \rightarrow B : K_{AB}\{A, g^a \ mod \ p\}$$

$$A \leftarrow B : K_{AB}\{B, g^b \ mod \ p\}$$

$$K_{session} = g^{ab} \ mod \ p$$

$$A \rightarrow B : K_{session}\{message, timestamp\}$$

$$A \leftarrow B : K_{session}\{message, timestamp + 1\}$$

List:

$$A \rightarrow S : \{\text{" } list \text{ "}, A, c_1\}_S$$

$$A \leftarrow S : K_A\{list \ of \ name, c_2\}$$

$$A \rightarrow S : \{\text{" } confirmed \text{ "}, A, c_2\}_S$$

LOGOUT

- Abort the command if server do not receive ACK

$A \rightarrow S : \{\text{" } logout \text{ "}, A, c_1\}_S$

Server send "A want to logout" to all clients except A, and clients comfirmed

Clients delete A connections

$S \rightarrow B : K_B\{\text{" } logout \text{ "}, A, c_2\}$

$S \leftarrow B : \{\text{" } confirmed \text{ "}, A, c_2 + 1\}_S$

Server confirmed all clients know A logout, and send confirmation back to A

$A \leftarrow S : K_A\{c_1 + 1, c_3\}$

$A \rightarrow S : \{\text{" } confirmed \text{ "}, A, c_3 + 1\}_S$

SESSION KEY PROTECTION

- Heartbeat:
  - Client send server heartbeat message every few seconds to keep connection alive, and prevent man in the middle attack
- Set expiration time for session key
- Timestamp:
  - Only receive message having timestamp within reasonable tolerance, disconnect and force to change session key if timestamp is over range.

EXTRA DESIGN

- DoS attack: if receive lots of message from the same address in few seconds, add the address to black list.
- Strengthen Weak Password: Bcrypt
- Perfect Forward Secrecy: Your session keys will not be compromised even if the private key of the server is compromised. Using diffie-hellman to generate unique session key only known by clients for each session.
- Message secrecy: message will be encrypted by AES256 using session key
- Endpoint hiding: DH + authentication allows anonymous connection