Instant Messaging System Design

Team: Tiny_Sniper
Designers: Guanyu Wang, Xiaoyuan Liu

Instant Messaging System Design

General design:

Authentication between client and server in the login process

Authentication between different clients when a client tries to communicate to another

Authentication between client and server in the logout process

In the system, server performs as KDC as well

Server stores:

- Hash^{128}(password | salt)
- RSA private key of the server

Client stores:

- RSA public key of the server

Login Authentication

User A input username and password into the workstation.
Client A and Server mutual authentication

$W_A \rightarrow$ Server: Login request
Server $\rightarrow W_A$: $[\mathrm{Hash}(R_1|\mathrm{salt}), \mathrm{salt}, \text{length of } R_1]_S$
$W_A \rightarrow$ Server: $K_{s\_pub}\{A, R_2, g^a \bmod p, \mathrm{Hash}^{128}(\mathrm{password} \mid \mathrm{salt})\}$, R_1, HMAC(K, M)
K = $\mathrm{Hash}^{128}(\mathrm{password} \mid \mathrm{salt})$
Server $\rightarrow W_A$: $[\mathrm{Hash}(R_2 \mid \mathrm{Hash}^{128}(\mathrm{password} \mid \mathrm{salt}))\{g^s \bmod p\}, \{R_2 + 1, R_3\}S_A]_S$

$S_A = g^{as} \bmod p$

$W_A \rightarrow$ Server: $\{R_3 + 1\}S_A$, HMAC(S_A, M)

List Operation

User A wants to get the list of users online:

A → S: S_A\{A, List request, TS\}, HMAC(S_A, M)
S → A: S_A\{List of online users, TS+1\}, HMAC(S_A, M)

TS is the timestamp of A generating the request.
This timestamp has nanosecond granularity

User to User Authentication

User A wants to communicate with User B:

$A \rightarrow KDC$: $S_A\{A, B, N_A\}$, HMAC(S_A, M)
$KDC \rightarrow A$: $S_A\{K_{AB\_tmp}$, IP_Addr_B, port_B, $N_A\}$, $S_B\{K_{AB\_tmp}, A\}$,
HMAC(S_A, M)
K_{AB_tmp}, temporary key for communication

$A \rightarrow B$: $S_B\{K_{AB\_tmp}, B\}$, $K_{AB\_tmp}\{g^a \bmod p, R_1\}$, HMAC($K_{AB\_tmp}$, M)
$B \rightarrow A$: $(K_{AB\_tmp}+1)\{g^b \bmod p, R_1+1, R_2\}$, HMAC($K_{AB\_tmp}$, M)

$K_{AB} = g^{ab} \bmod p$
K_{AB}, Session key for exchanging messages
$A \rightarrow B$: $K_{AB}\{R_2+1, R_3\}$, HMAC(K_{AB}, M)
$B \rightarrow A$: $K_{AB}\{R_3+1\}$, HMAC(K_{AB}, M)

Logout

WA → Server : S_A\{Logout Request of A, TS_A\}, HMAC(S_A, M)

TS is the timestamp of A generating the request. This timestamp has nanosecond granularity
Server compare the TS received with the current timestamp, if within a acceptable difference, grant the request.

Server → WA : S_A\{Logout Response of A, TS_A+1, TS_S\}, HMAC(S_A, M)
WA → Server : S_A\{TS_S\}, HMAC(S_A, M)

S_A expires, and A gets removed from the online users list.

All timestamps have nanosecond granularity

Vulnerability to DOS attack

In login authentication, client needs to send hashed password and salt encrypted by server's public key. There is a possibility that server needs to decrypt large amount of data using private key. In our design, client needs to send the answer to the challenge and server will check the challenge first before decrypting files.

Online and offline attacks

Online attack:
5 wrong trials will lead to lock of user account for 30 minutes.

Offline attack:
Assume that the intruder has already stolen the database of the server and got hash values of password and salt of every user.
Since the server stores hash^128 of the password and salt, it leads to very low possibility of brute forcing the password using dictionary.

End-Points Hiding Issue

In our design, every message containing identity of user is encrypted. The list of users, their IP addresses and ports are also encrypted. Since the key is secure enough, the intruder can not know the specific user by getting the IP address of packets.

No Trust KDC

User A wants to communicate with User B:

$A \rightarrow KDC$: $S_A\{A, B, N_A\}$, $HMAC(S_A, M)$
$KDC \rightarrow A$: $S_A\{K_{B\_pub}, IP\_Addr\_B, port\_B, N_A\}$, $S_B\{K_{A\_pub}, A\}$, $HMAC(S_A, M)$
$A \rightarrow B$: $[S_B\{K_{A\_pub}, B\}, K_{B\_pub}\{g^a \bmod p, R_1\}]_A$
$B \rightarrow A$: $[K_{A\_pub}\{g^b \bmod p, R_1+1, R_2\}]_B$

$K_{AB} = g^{ab} \bmod p$
K_{AB}, Session key for exchanging messages
$B \rightarrow A$: $K_{AB}\{R_2+1, R_3\}$, $HMAC(K_{AB}, M)$
$A \rightarrow B$: $K_{AB}\{R_3+1\}$, $HMAC(K_{AB}, M)$

Compared to the design in which users trust KDC:
This design has potential vulnerability of DOS attack, since there are two exchanges requiring encryption and decryption using asymmetric keys.