Secure Chat Design Document

Christophe Leung • Yang Yang

Client Authenticates Server
$A \rightarrow C$: $\{ M_A, N_A, K_A, T_A \}K_C$
$C \rightarrow A$: $\{ N_A, T_C \}K_A$

Server Authenticates Client (User Login)
$A \rightarrow C$: $\{ ID_A, PW_A \}K_C$
$C \rightarrow A$: $\{ M \}K_A$

Client to Client Authentication (Needham Schroeder)
$A \rightarrow C$: $\{ ID_A, ID_B, N_A, T_A \}K_C$
$C \rightarrow A$: $\{ K_s, portB, ID_B, N_A, \{ K_s, ID_B, T_C \}K_B \}K_A$
$A \rightarrow B$: $\{ P_A, ID_A, T_A, N_A, \{ K_s, ID_B, T_C \}K_B \}K_B$
$B \rightarrow A$: $\{ \{ N_A, T_B \}K_A \}K_S$
$A \rightarrow B$: $\{ \{ M_A, T_A \}K_B \}K_S$

Client to Client Communication
$A \rightarrow B$: $\{ \{ M_A, T_A \}K_B \}K_S$
$B \rightarrow A$: $\{ \{ M_B, T_B \}K_B \}K_S$

ID = username
PW = password
P = port
T = timestamp
N = nonce
M = message
K = key

A = Client A
B = Client B
C = Server
S = Client-to-client Session

Our chat client makes use of multiple checks to ensure a secure chat environment.
The client first authenticates the server by sending a nonce and the client's own public key, all encrypted using the server's public key. If the server is authentic, it will decrypt the message and return the nonce in a timely manner.

The client then logs in, again using a nonce to verify its messages' integrity. The server authenticates the client and replies with an "success" message encrypted with the client's public key.

The client then initiates a chat by typing <message> *username*, which uses Needham Schroeder to establish a secure client-to-client chat session.