Susmitha Manda and Lucas Hagel
Secure Chat Protocol Design v1.0

We use parts of the Expanded Needham Schroeder protocol and we add a few tricks to make it viable for an implementation. Alice and Bob are general usernames in the system. In implementation, messages will be transferred over TCP using Java sockets; in the protocol descriptions N_{auth}/etc. will be 1024-bit random numbers.

Situations: Establishing a username, communicating with a buddy

Login

We authenticate Alice to the server.

Alice → Server: {Alice, SHA256(password), N_{auth} } $K_{public, server}$

K_{AS} = SHA256(N_{auth})

Alice calculates K_{AS}

Server → Alice: {BuddyList | hash(BuddyList) } K_{AS}

Communicating with somebody on buddy list

Server gets Bob's permission to talk to Alice, then passes the connection to Alice and Bob. Alice authenticates Bob at the outset to say she wants to communicate by signing with the server's public key because only the server can decrypt that message. We are assuming Bob is already online; if not, then notify Alice and restart the session.

Alice → Server: {Bob|hash(Bob) }K_{AS}

If Bob properly responds to the challenge, then the server passes connection to Alice and Bob

Server generates K_{AB} = SHA256(N_{Serv})

Ticket = {K_{AB} | hash(K_{AB}) }K_{BS}

Server → Alice: {K_{AB} , ticket | hash(K_{AB} , ticket) }$K_{AS,}$

Alice → Bob: ticket , "let's talk" //authenticates Alice to Bob

Susmitha Manda and Lucas Hagel
Secure Chat Protocol Design v1.0

Alice ←→ Bob: {message|| SHA256(message) }K_{AB} //if Bob doesn't send a comprehensive message, then he wasn't able to decrypt K_{AB}

Message integrity is provided by hashing the message with the last nonce used in the key exchange. This makes it so that if K_{AB} has been compromised and the adversary does not know $N_{challenge\ to\ Alice}$, Alice or Bob may be able to detect foul play. Confidentiality is provided by AES encrypting with K_{AB} and using CBC.

Can hard code buddies,
Usernames, password, ~10 users each 3-5 buddies