DefCom

Cryptographic Multi-Chat Application Design Document

Thavy Thach
Skylar Levey

Overview of System Architecture

This application is a multi-party, secure chat room for which a server will be used for communication between individuals. Each individuals has a password protected identity which they will use to authenticate themselves when they log in to the chat service. Once authenticated (by password) the client is able to either make there own new chat room, or join a pre-existing one. In the case of the former the client specifies which other users should be able to access and participate in the conversation upon creation of the given chat room. Once a user is a member of a chat, they are able to join it and then send encrypted messages to other members of the chat room. Clients can also exit the chat service from the main menu, and when they do they will stop receiving messages from any chat they are currently participating in.

Attacker Model(s)

1. Attacker has full channel control.
2. Attacker cannot break cryptographic primitives.
3. Attacker is able to replay messages.
4. Attacker is able to impersonate active users.
5. Server can leak keys.
6. Server can be compromised.

Security Requirements

First of all, it is important that a client's log in to the server is somehow secured by encryption as if a user was to simply send their password in plaintext to the server it would be very easy for the attacker to eavesdrop and then impersonate them.
Secondly, due to the fact that the Attacker has complete channel control it is important to make sure that messages have not been tampered with, so using some form of a MAC or Digital Signature as message content insurance is important.
Because modification of messages will (hopefully) be impossible for attackers, we must also be prepared to protect against replay attacks by using timestamps (as the chat service is implemented locally, meaning the clock should be reliable) to make sure that after a message is received the same user will not accept an identical message.
Finally, it is very important to make sure that an attacker is unable to read messages sent between clients in a chat room (this is a *secure* chat room after all) so using a strong encryption on the messages sent between users is a necessity.

Cryptographic Primitives

For Encryption we will be using AES in CBC-mode with 32-bit keys and IVs generated by cryptographic pseudo random number generators
For Message Authentication we will be using SHA-256 with 32-bit keys
For Digital Signatures we will uses HMAC hashes encrypted with the Sender's private key
For Replay Protection we will be using Timestamps, both the Client and the Server will discard any message that has a Timestamp more than 30 seconds old

Authentication Upon Login using Password

When a Client logs in they send a message to the server with their encrypted Username and password which the server checks and if valid, logs them in

Authentication Upon Joining a Chat Room

Firstly, we will implement a server check that prevents anyone but the invited participants from joining a chat room. In order to join a chatroom you must both know the chat room ID and be confirmed in the chat room by the server.

When a client logs into a chat room there are two possible scenarios:

1. They are the first to join the chatroom
2. There are other participants in the chat room.

In the first case the client will create a fresh chat session key with a PRNG and then if they decide to write anything it will be encrypted and saved to the chat log

In the second case, they client logging in will select a random currently active member of the chat room and ask them for a key, to which they will respond with the current key enabling the new user to decrypt previous messages and then participate.

Timestamps will be included when messages are decrypted locally to serve as sequence numbering explicitly.

Authentication Upon Login

CLIENT sends Signifier and Authentication Request, where the signifier dictates the start of the login protocol.

$$(Signifier \mid Timestamp \mid Public\ Key\ of\ Client \mid ENC^{Server\ Public\ Key}(Username,\ Password)\ )\ Signature_{Client}$$

SERVER sends a response back to...

$$(Timestamp \mid "You\ have\ Successfully\ Logged\ in!!")\ Signature_{Server}$$

Fig. 1.0… Authentication Request using Key Exchange Protocol

Service Request

SERVER sends available chat rooms and waits for user action

$$(Timestamp \mid "Available\ chat\ room : ...What\ would\ you\ like\ to\ do...)\ Signature_{Server}$$

CLIENT responds with their option

$$(\ Timestamp \mid\ < make\ a\ new\ room,\ enter\ a\ room,\ or\ exit\ service >)\ Signature_{Client}$$

Fig. 2.0… Service Request

Chat Room Creation *(make a new room option)*

CLIENT sends necessary information to the server

$$(\ Timestamp \mid Client\ Username \mid Other\ included\ Users)\ Signature_{Client}$$

SERVER then begins logging the chat room and will send it to included users during service requests, server then sends confirmation

$$(\ Timestamp \mid "Chat\ room\ with\ < included\ Users >\ created)\ Signature_{Server}$$

Fig. 3.0… Chat Room Creation

Entering A Chat room *(enter a room option)*

CLIENT sends choice of chatroom to server

$$(Timestamp \mid Client\ Username \mid Chatroom\ name)\ Signature_{Client}$$

SERVER responds with the current number of participants in the chat room

$$(Timestamp \mid Current\ Number\ of\ Participants)\ Signature_{Server}$$

if the user is not an eligible member of this chat room server responds with

$$(Timestamp \mid "You\ are\ not\ a\ member\ of\ this\ chat\ room")\ Signature_{Server}$$

CLIENT then responds in one of two ways:

<u>**1**</u>: if this client is the first to join the chat room, the client will locally create a new chatroom session key that will be used for this session:

$$(Timestamp \mid "Confirmation/chat\ room\ is\ active"\ )\ Signature_{Client}$$

<u>**2**</u>: If there are other members currently in the chat room the client must initiate a key exchange protocol with a randomly selected, currently active member of the chat room in order to get the session key and start participating in the conversation

$$\begin{aligned}&(Timestamp \mid Chat\ Room\ ID \mid Client\ Username \mid Public\ Key\ of\ Client\\ &\qquad \mid "requesting\ key\ for\ this\ chat\ session"\ )\ Signature_{Client}\end{aligned}$$

Server then forwards the message to a randomly selected, currently active member of the chat, They (**Active Client)** then respond with

$$\begin{aligned}&(Timestamp \mid Chat\ Room\ ID \mid Active\ Client\ Username \mid Client\ Username\\ &\qquad \mid ENC^{Client\ Public\ Key}("session\ key"\ )\ Signature_{Active\ Client}\end{aligned}$$

Server then checks that the Username checks out for the chat room and then forwards it to the original client. With the chat room session key the user can then tell the server to send them previous messages messages and start sending them new ones as well

$$(Timestamp \mid "Confirmation/chat\ room\ is\ active"\ )\ Signature_{Client}$$

Finally once the Client has joined the chat room they can start participating in the conversation with messages structured like this

$$(Timestamp \mid Chat\ Room\ ID \mid Client\ Username \mid ENC^{Chat\ Room\ Session\ Key}(\text{"}message\text{"}))\ Signature_{Client}$$

Fig. 4.0… Entering a Chat Room

Exiting a Chat Room *(exit service option)*

CLIENT sends service request to quit the server

$$(\ Timestamp \mid Client\ Username \mid Chatroom\ name\ )\ Signature_{Client}$$

SERVERS accepts service request to quit by outputting to the chat <Client Username> left <Chatroom name>, updates current number of participants in the chat room, and maintains chatroom session key among current participants.

$$(\ Timestamp \mid \text{"}Client\ < Username\ > left\ Chatroom\ name\text{"} \mid$$
$$Chat\ room\ with\ < included\ Users >\ -\ Client\ Username\ )\ Signature_{Server}$$

Fig. 5.0… Exiting a Chat Room