Secure Chat System

Group Members: Nathan Sumpter, Stephen Shinn, Jasper Chen, Allen Sarmiento, Hector Medina

Abstract

This project implements a secure chat system where one or more users can join together in a group chat. Each user must be registered with the chat server through a username and password. Security is established through the use of a symmetric key between all users within the chat.

Introduction

Our project delivers secure exchange of chat message for all parties involved. Users log into a chat system and can select another user with whom they want to participate in a conversation with. Users can check the authenticity of messages they received through the implementation of digital signatures.

Design

This chat system is designed to consist of the following:

- Storing username and hashed passwords in a database
- Directing user to chat interface upon valid username and password
- Input field for user to type in a message
- Radio buttons for user to choose to sign the message with RSA or DSA
- Send button to be able to send the message to another user
- Clickable icons to select a person to initiate a chat with
- Online and offline indicators to show which users are available and which are not available to hold a chat session
- Logout button to log the user out

Use Cases

Validate User Login	
Use Case #	1
Objective	Validate username and password to determine whether the user should be logged in or not
Priority	Very High
Primary Actor	System
Flow of Events	
1 Basic Flow	1. User opens up the application 2. User enters in username 3. User enters in password 4. User hits the log in button 5. Username and password get compared with the database values

	6. Direct user to chat system if login is valid
2 Alternate Flow	NONE
3 Includes	NONE
4 Pre-condition	Username and password must be stored in database
5 Post-condition	User is in the chat system interface

Send Message	
Use Case #	2
Objective	User sends a message to another user
Priority	Very High
Primary Actor	User
Flow of Events	
1 Basic Flow	1. User selects an online user to chat 2. User types a message in the input field 3. User hits the send button 4. The message is send to the user
2 Alternate Flow	NONE
3 Includes	NONE
4 Pre-condition	Other users must be online
5 Post-condition	The message is successfully sent

Receive Message	
Use Case #	3
Objective	User receives a message from another user
Priority	Very High
Primary Actor	User
Flow of Events	
1 Basic Flow	1. Both users trying to communicate are logged in 2. One user types a message in the input field 3. The user then hits the send button 4. The other user receives the message
2 Alternate Flow	NONE
3 Includes	NONE
4 Pre-condition	Other users must be online
5 Post-condition	The recipient receives the message

Setting status as online	
Use Case #	4
Objective	When a user is logged in, their status is set to online
Priority	High
Primary Actor	User
Flow of Events	
1 Basic Flow	1. User types in username and password 2. User gets verified and proceeds to chat application 3. User's icon is green, indicating online
2 Alternate Flow	NONE
3 Includes	NONE
4 Pre-condition	Username and password in registered in the database
5 Post-condition	User's icon on the right is green

Setting status as offline	
Use Case #	5
Objective	When a user hits log out, their icon changes to pink, indicating offline
Priority	High
Primary Actor	User
Flow of Events	
1 Basic Flow	1. User is in chat application and has online status 2. User clicks log out button 3. User is directed back to login page
2 Alternate Flow	NONE
3 Includes	NONE
4 Pre-condition	User has signed in through the login page User's icon is green, indicating online
5 Post-condition	User's icon is pink, indicating offline User is directed to the login page

Logging Users Out	
Use Case #	6
Objective	Log users out when they hit the log out button
Priority	High
Primary Actor	User

Flow of Events	
1 Basic Flow	1. User logs in 2. User is validated and is in the chat system 3. User hits the log out button
2 Alternate Flow	NONE
3 Includes	NONE
4 Pre-condition	User must be logged in User must be in the chat interface
5 Post-condition	User is logged out and directed to the login page

Sign message with RSA or DSA	
Use Case #	7
Objective	Create a digital signature with RSA or DSA
Priority	High
Primary Actor	System
Flow of events	
1 Basic Flow	1. User logs in 2. User is validated and is in the chat system 3. User types in a message and chooses either RSA or DSA 4. User hits the send button 5. System creates the digital signature with the user's choice
2 Alternative Flow	NONE
3 Includes	NONE
4 Pre-condition	User must be logged in User must be in the chat interface User must select RSA or DSA User types a message and hits send
5 Post-condition	System digitally signs the message

Security Protocols

The cryptographic protocols used in this project include:

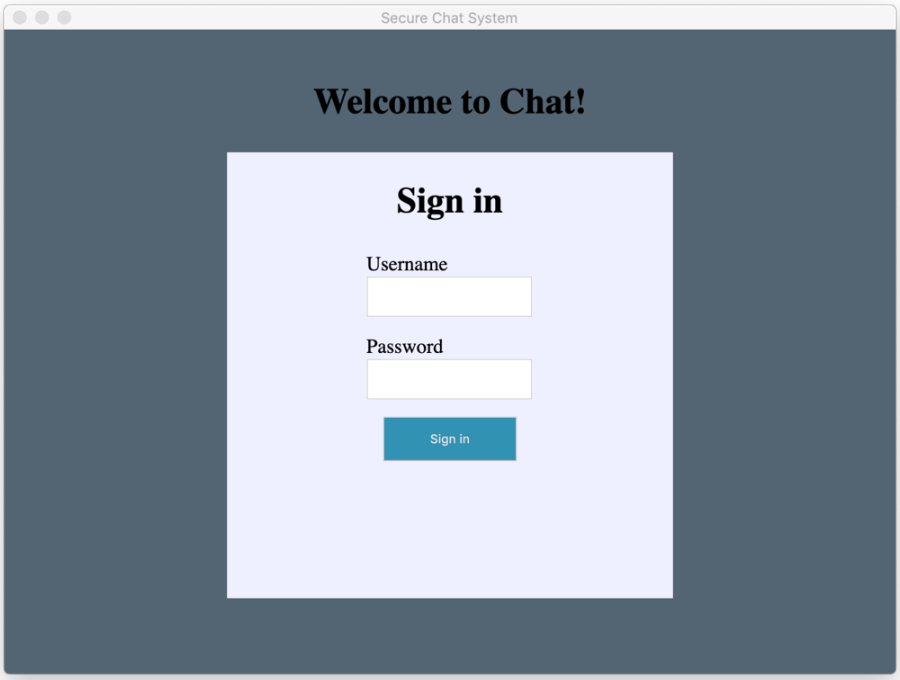
- Hashing and salting of passwords to achieve confidentiality in case of the database being compromised
- Private and public key creation to support authentication through the use of digital signatures to sign a message
- Symmetric keys to offer communication between two parties
- Digital signatures with either RSA or DSA to ensure authentication of the sender

Using these security protocols, we can ensure a secure communication in a chat application.

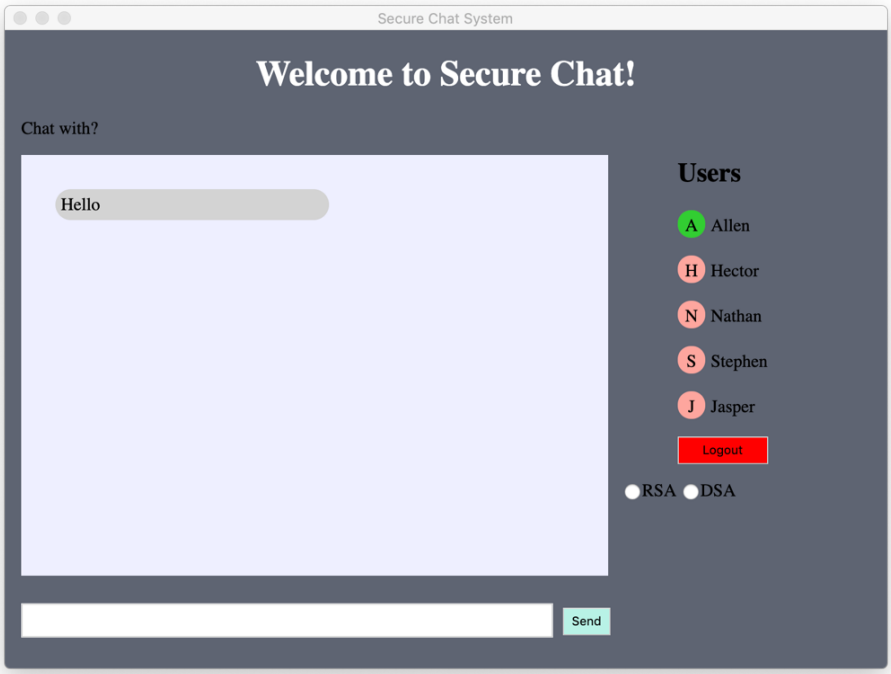
Implementation

This project runs with Electron on client side and Python on the server side. We chose to use Electron on the client side, because we could create the interface using HTML, CSS, and JavaScript in hopes that designing the interface would be easier and less time consuming. Python is used on the server side because it was the preferred language to use. Some libraries that are included are: jQuery to handle ajax requests, crypto for creating digital signatures, and bcrypt to hash and salt passwords. Postman is another tool that was utilized to figure out the start and end points for communication between the client and server.

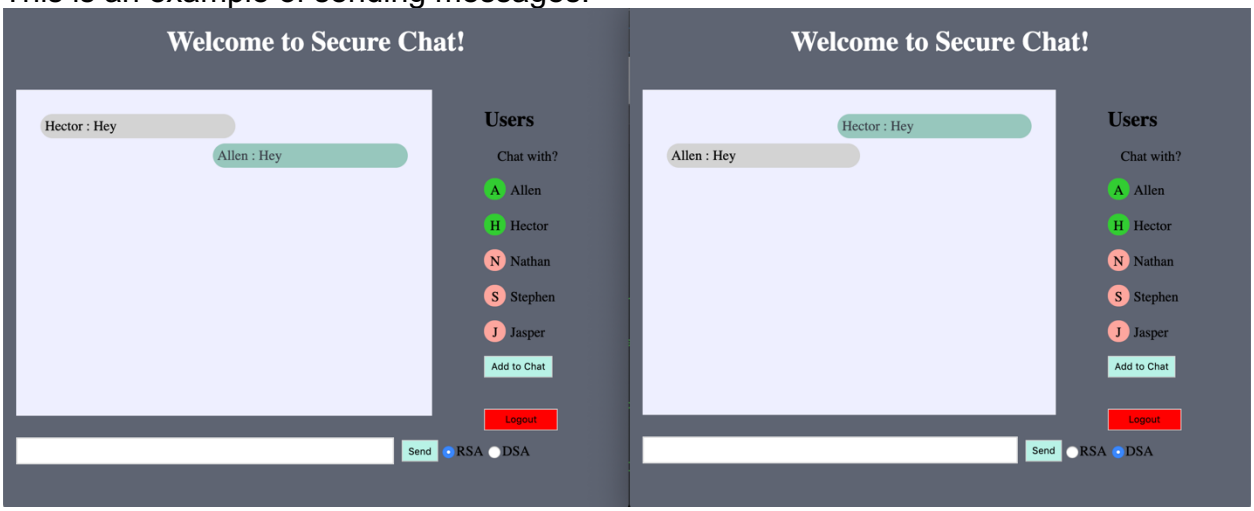
Here is a screenshot of the login page:



This is a screenshot of the chat system interface:



This is an example of sending messages:



Conclusion

This project taught us a lot about how to make chat systems secure. We learned to apply our knowledge of digital signatures to ensure authentication when sending and receiving messages. We learned why it is important to hash and salt passwords due to vulnerabilities without doing so and had the chance to be able to apply that knowledge in an application. When dealing with sensitive information, it is important to implement secure protocols.