Brett Lockhart
Gary Nunez
Gary’s Loaf of Bread

Application Properties

This application will be a messaging application that allows text communication between two users. The app will not support group chat. Our application will provide a secure way to pass simple text messages between users by using asymmetric encryption and certificates.

Our messaging application will be a web app built with HTML, CSS, JavaScript and PHP. We will implement a RESTful web server written in PHP, using AWS to host the server.

Assets and Stakeholders

Assets:

- Messages and information sent between users.
- Database

Stakeholders:

- Users of the application

Adversarial Model

Online Insider

An adversary within the system will attempt to attack, can be through the server of other means, but they would have to get into the system first. To do this they would have to get through the SSL which make sure all that every message that goes through the server has a valid certificate. Lastly if an insider steals information from the database through the server, they would not be able to do anything with it because the server has no access to the private keys of the clients so all the data will still be encrypted.

Online Outsider

This adversary is one that will attempt to intercept messages going to or coming from the server. In order to protect ourselves from this type of attack, we will be implementing AES encryption

Offline Outsider

An adversary that doesn’t tamper with the system but instead does engage in workstation hijacking where the adversary can steal a device with login credentials. There is not protection against this attack.

Possible Vulnerabilities

A current vulnerability is that we do not effectively know how to send the private keys of the clients to each other without the server being aware of it. If an insider adversary has access to the server, they can retrieve the keys to decrypt the message.

Possible Related Previous Work

WhatsApp, Signal

Complete Description of Solution

Our solution must allow a user to send a message to another user without the possibility of the message being accessed or tampered with before it reaches the intended user. To do this, our system will have 2 clients, each with its own public key that can be passed to the other user. They will be able to encrypt messages with these keys and send those messages to each other. To get to their destination, the messages must first pass through the server and the server will hold the message until it is ready to be received by the recipient. Once the recipient receives the message, the client will decrypt the message with its private key and then be able to read it and work on a response.

To achieve End to End Encryption in our solution, it must prevent adversaries from accessing the messages. This can be done by using a combination of Public/Private key cryptography and SSL. The public/private key cryptography will use the public key to encrypt the private key and allow one client to send their public key to the other client so they could encrypt messages that only they can read. To get the messages to the recipient safely, SSL will ensure that only authorized users can access the messages through the use of certificates.

Full rigorous analysis

Confidentiality

Our implementation will maintain confidentiality for our users. They use this application because they want their conversations to be private. Our use of the TLS 1.2 protocol with AES encryption and certificates, users can be sure that their message is indeed confidential. Only the recipient of the message with the correct Secret Key will be able to decrypt the message. That means that the only person who can read the plaintext of a given message is the sender and the receiver.

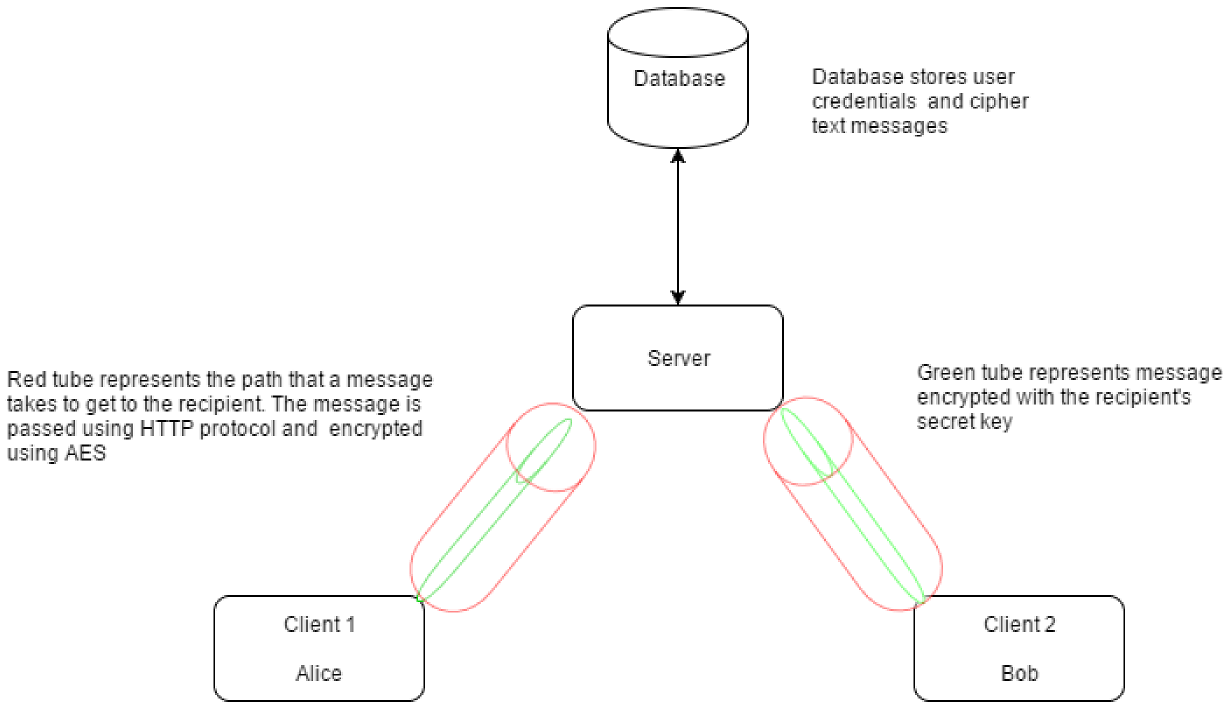
Integrity

Our RESTful server will be implemented in a way that protects the integrity of the system. It should be protected against code injections or modules being manipulated.
The integrity of messages will be maintained
**Read about establishing green tube*

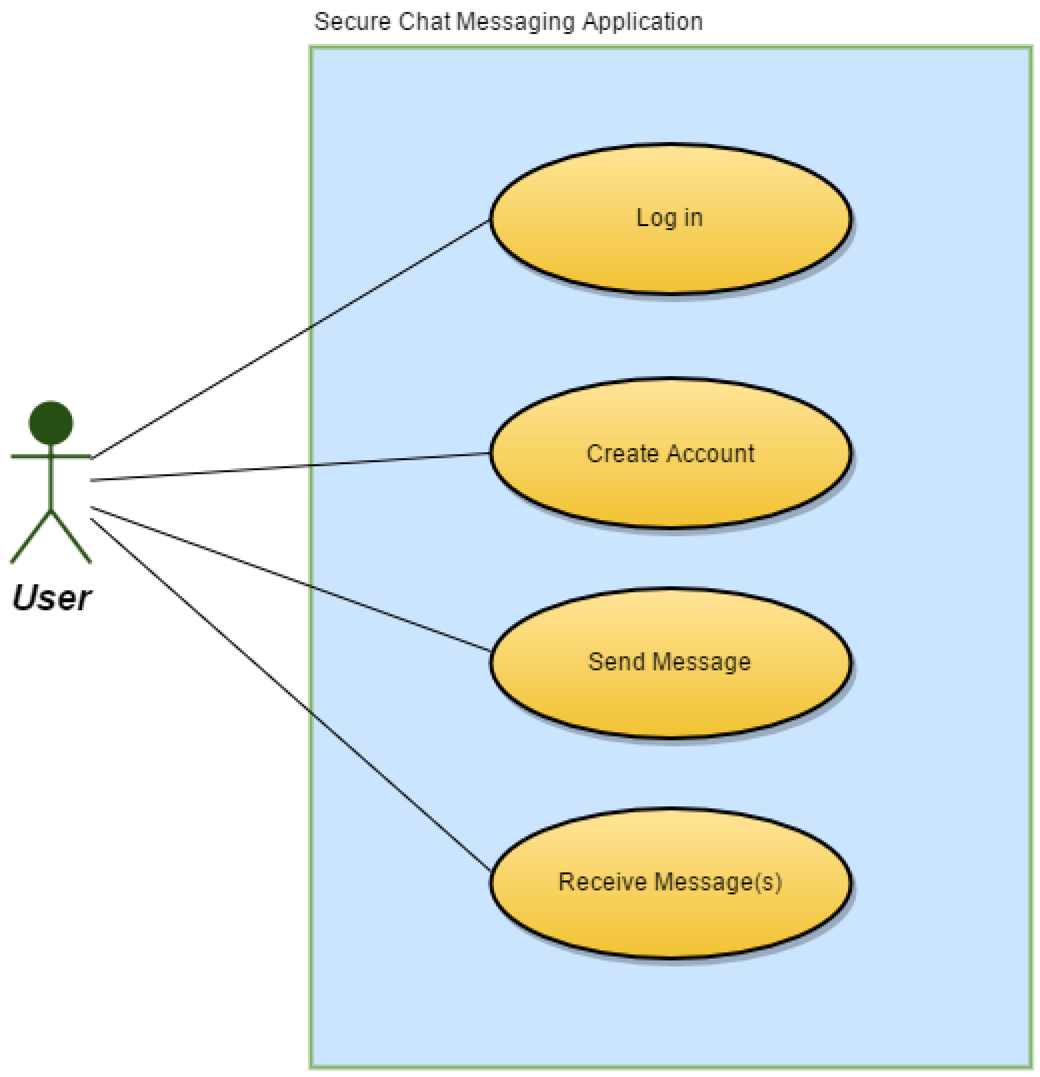
Authentication

The system will handle authentication by using a credential service provider to prove a user's identification. Once the credential service provider is satisfied, the user will create a unique username and be provided with a token that allows the SSL certificate to recognize the user. Then the system will allow the user to send and receive messages.

System Diagram


Database
Database stores user credentials and cipher text messages
Server
Red tube represents the path that a message takes to get to the recipient. The message is passed using HTTP protocol and encrypted using AES
Green tube represents message encrypted with the recipient's secret key
Client 1
Alice
Client 2
Bob

Use Case Diagram



Use Cases

Log In

Objective	Allow User to Log In
Priority	High
Source	Client, server
Actors	Client
Flow of events	Basic Flow: 1. User navigates to website 2. User enters username and password 3. Server queries the database to validate credentials 4. User enters the main screen Alternate Flow: 1. User Navigates to Website 2. User clicks to create new account 3. User is taken to Create Account Screen 4. User enters desired username and password 5. After entering credentials, account is created and user is taken to main screen
Precondition	User is logged out of the application.
Post condition	User is logged in to the application
Notes/issues	none

Send Message

Objective	Allow user to send a message
Priority	High
Source	Client
Actors	Client, server
Flow of events	1. User selects a contact to send message to 2. User types the message they wish to send 3. User presses the send button 4. The server stores the message in the database until the recipient is ready to receive the message

Precondition	User has not sent any messages.
Post condition	User has sent a message.
Notes/issues	none

Receive Message

Objective	Allow user to receive messages
Priority	High
Source	Client
Actors	Client, server
Flow of events	Basic flow: 1. Server checks if the user is logged in 2. If the user is logged in, server sends message to recipient. Else if user isn't logged in, the server will continue to hold the message until the user is able to receive it.
Precondition	User is logged in.
Post condition	User has received messages from the server.
Notes/issues	none