Secure Chat Web Application

Brett Lockhart
Gary Nunez

Platform and Languages

- Project will be implemented as a web application
- Accessible to most platforms with an internet browser and JavaScript enabled
- The current language considered for implementation is PHP using the Laravel Framework.

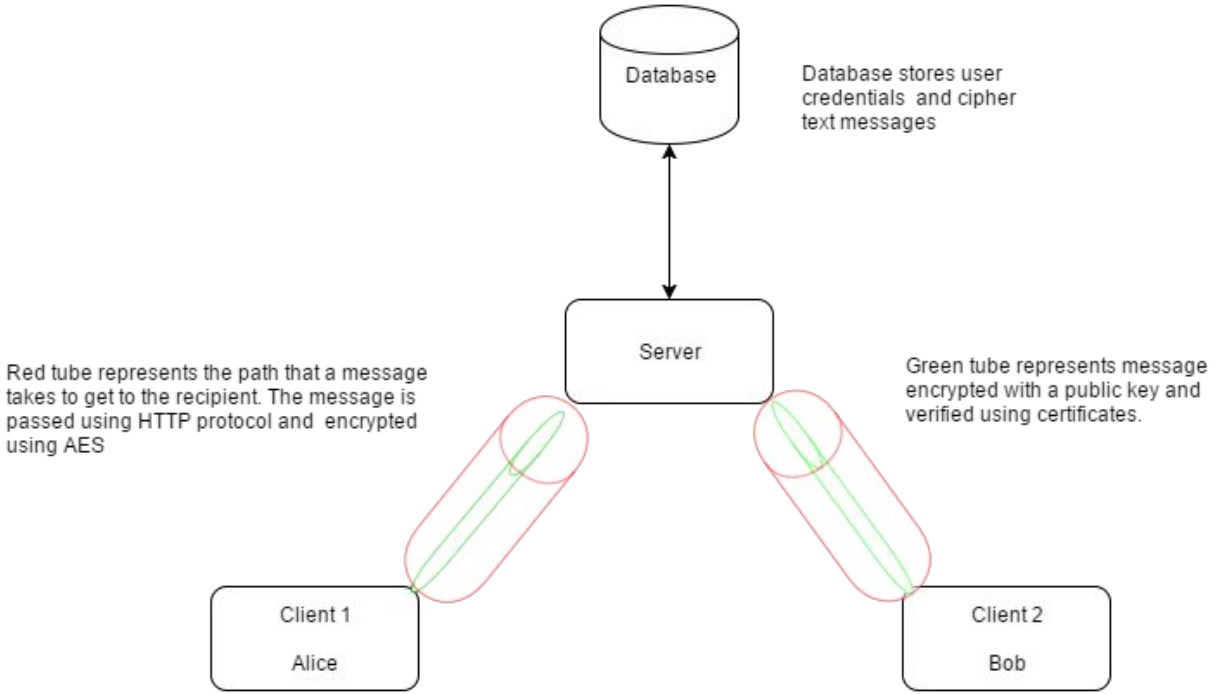
Solution

- System will allow 2 clients to send messages to each other
- Will use E2EE encryption to secure messages
  - Will prevent server from viewing private data
  - Will protect messages from being intercepted or tampered with by outsiders
- When a message encrypted and is sent, the server will store the cipher text in a database until it is ready to be received
- Recipient can decrypt the cipher text using their private key

E2EE (End to End Encryption)

- Prevents adversaries from accessing private data.
- Achieved by using **public/private key cryptography**
  - Each client has their own set of keys
  - Public and private key
  - Alice sends public key to Bob
  - Bob can now send secure messages to Alice that only she can decrypt
- Ensures that data is secure
- TSL will make sure that only authorized users can access data

System Diagram



Adversarial Model

Online insider

- 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 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 no protection against this attack.

Analysis

Confidentiality

- Our use of the TLS 1.2 protocol with 256 bit AES encryption and certificates, users can be sure that their message is confidential. Only the recipient of the message with the correct Secret Key will be able to decrypt a message that is desired for them.

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 also be maintained

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.