Roadmap for Com-Sec

Registration Panel

Users can register on the platform by providing the following details:

1. Username
2. Email address
3. Password

The password is encrypted using a secure algorithm before being stored in our NodeJS server connected to a MongoDB database. Upon successful registration, the server assigns a unique ID to the device and generates three keys:

- (a) Key for encrypting the password.
- (b) A public key pair for calculating the session key.
- (c) Symmetric storage key for encrypting/decrypting local storage, which contains the contact list, chat history, and key store.

Login Panel

To log in to the application, users need to provide their:

1. Email
2. Password

If the server confirms that the credentials are authentic, a JSON Web Token (JWT) is created and sent to the client to be stored. When the client makes a request later, the JWT is passed with the request. The server verifies the JWT, and if it is valid, the request is processed.

Firebase Cloud

Firebase Cloud Messaging (FCM) is a service that facilitates messaging between mobile applications and server applications. It is a free service that allows sending lightweight messages from the server to the devices whenever new data is available.

1. The application connects to the FCM server and registers itself.

2. Upon successful registration, FCM provides a registration token to the device. This registration token uniquely identifies each device.
3. The application sends the registration token to the server, which stores it in the MongoDB database.

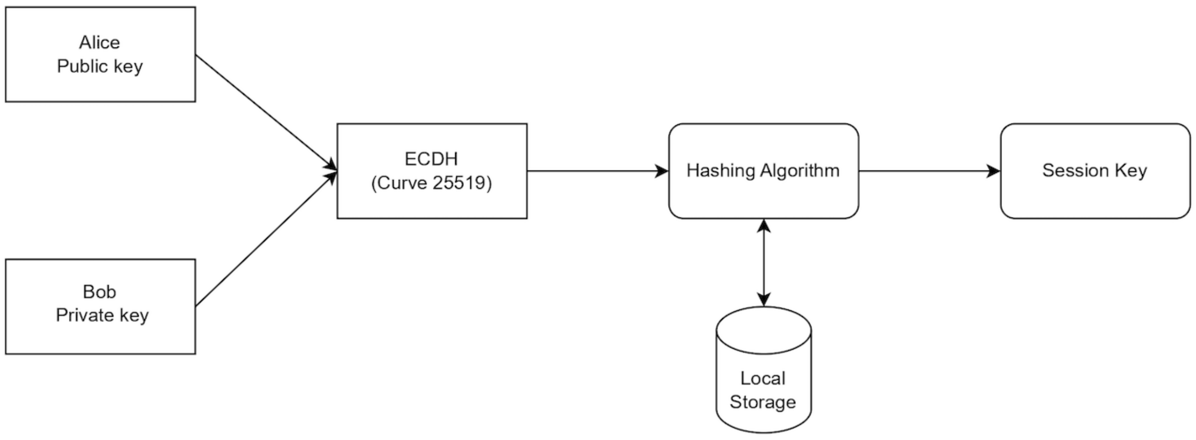
Peer Network

When Alice sends a request to Bob, the server stores Alice's public key. If Bob accepts the request, his private key is fetched, and combined with Alice's public key to calculate a session key using Elliptic Curve Diffie-Hellman (ECDH) over the Curve25519. The results are hashed using a specified hash algorithm, and the session is stored in local storage. The acceptance, along with Bob's public key, is then sent to the server to be delivered to Alice. Alice uses her private key and Bob's public key to calculate the session key, which is stored in local storage for later use.

Communication

When a message is typed, the application encrypts the message using the specified encryption algorithm to encrypt the message body and Poly1305 to compute a Message Authentication Code (MAC). Each message has its own separate key and nonce. After encrypting the message, it is encrypted again using the recipient's session key before being sent to the server.

After the message is received from FCM, the MAC of the encrypted message is calculated and compared with the received MAC to verify the integrity of the message. The message body is then verified using the same steps. Once the key and nonce to decrypt the message are known, the message is decrypted, stored in local storage, and displayed to the recipient.



Local Storage

Data is stored locally in the application using the Realm database, which is a lightweight and fast mobile database that supports cross-platform use. Encrypted data is protected from unauthorized access and is accessible only with the correct encryption key. Realm uses the AES-256+SHA2 algorithm and a 64-byte key for encrypting storage.

Server-Side Implementation

The server-side implementation relies on Node.js and a MongoDB database.

Step 1: Start by running the MongoDB connection and then run Node.js from the Command Prompt. The server is now ready to receive client requests.

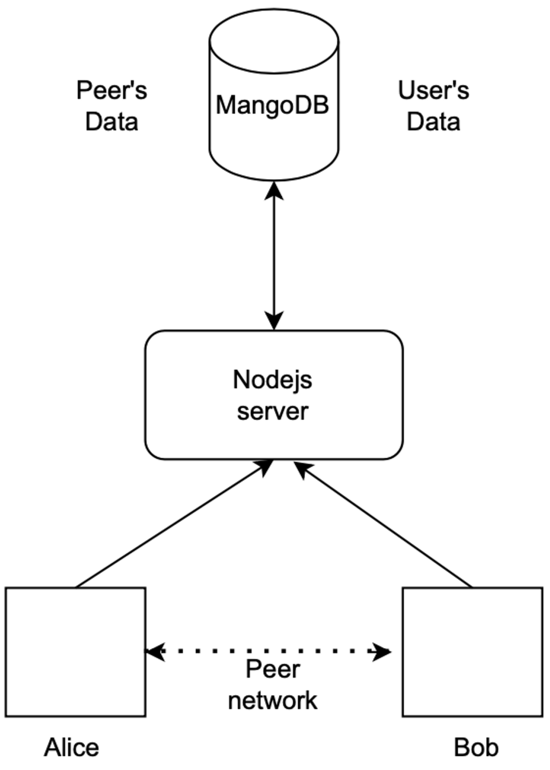
Step 2: When the client sends a request, the server receives the HTTP request in JSON format and parses it.

Step 3: The HTTP request is compared with the base path. If it matches, it is handed to the Express framework.

Step 4: Express receives the HTTP request and routes it to the specific endpoint that matches it. If no routes match, an error is displayed in the Command Prompt. Otherwise, the request is forwarded to the controller that handles the required function.

Step 5: A request is made to the MongoDB database using Mongoose for processing the function.

Step 6: When the data is fetched from the MongoDB database and the required operations are completed, Node.js receives the response and sends it to the client.


Peer's Data
MangoDB
User's Data
Nodejs server
Peer network
Alice
Bob