ChatSignal

Che-An(Andy) | Corbin | David | Jason | Nathan

The Problem: Secure Messaging

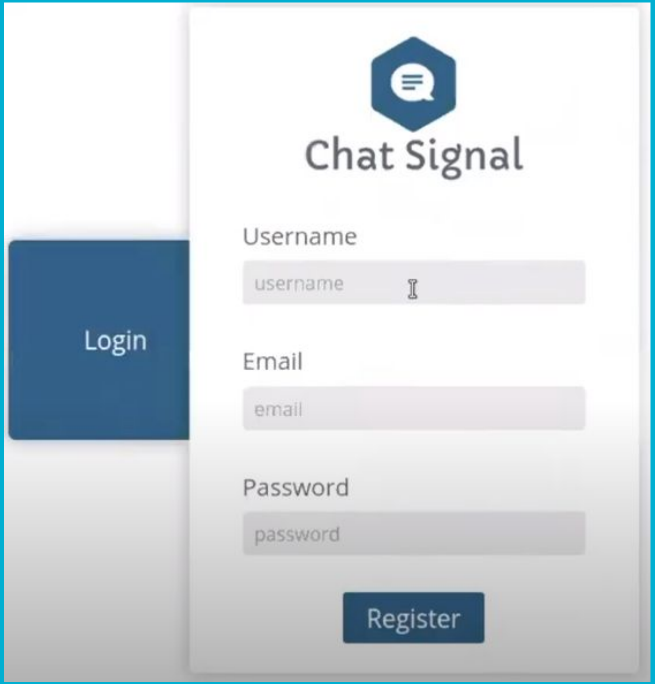
- People often assume that the discussion they have online are private, but due to a lack of security on the platforms they use, the contests of these discussions can be leaked.
- Some platforms even analyse user messages and sell information to advertisers.
- Leaked information can lead to financial, social, or even legal troubles.
- User with/without technical skills will not be able to verify if their message is actually encrypted as tech companies claimed.
- ChatSignal offers a platform where privacy and security come first.

Summary: Register/Login

- Salt & Hash Password (bcrypt)
  - Salt generates random values
  - Hash plain password with salt
- City Verification
  - If location does not match the location from the previous login attempt, the user is prompted to verify their email
- AWS services: API gateway, Lambda, DynamoDB

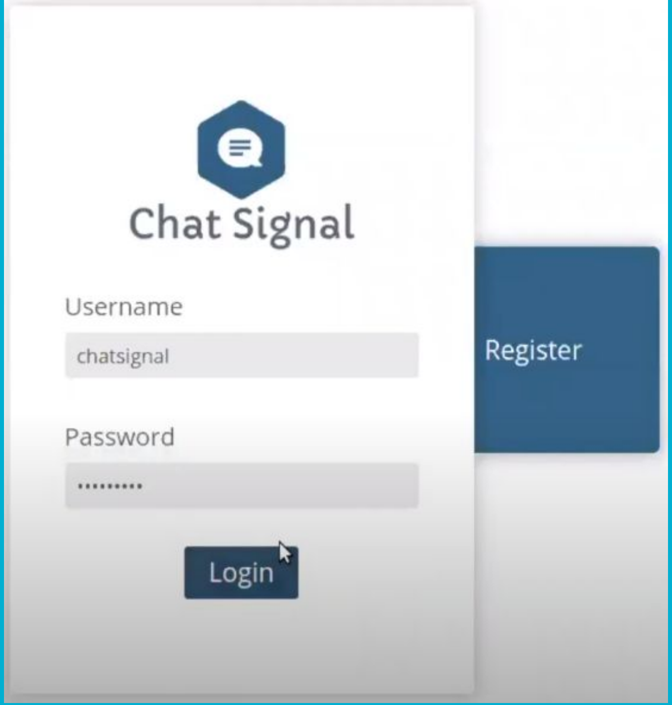
Signing Up

- User is asked to fill out their username, email, and password
- No duplicate username is allowed
- Password must contain a mix of uppercase, lowercase, and numbers
- User's city will be retrieved from IP address and stored with user info.
- If registration is successful, user will be sent to login page



Logging in

- User types in credentials
- Incorrect username/password combo denies login
- If entry in database, sent to the home page of our chat app
- If location does not match the stored location, user is prompted for email verification
    - New location is then sent from the user side and updated in our database

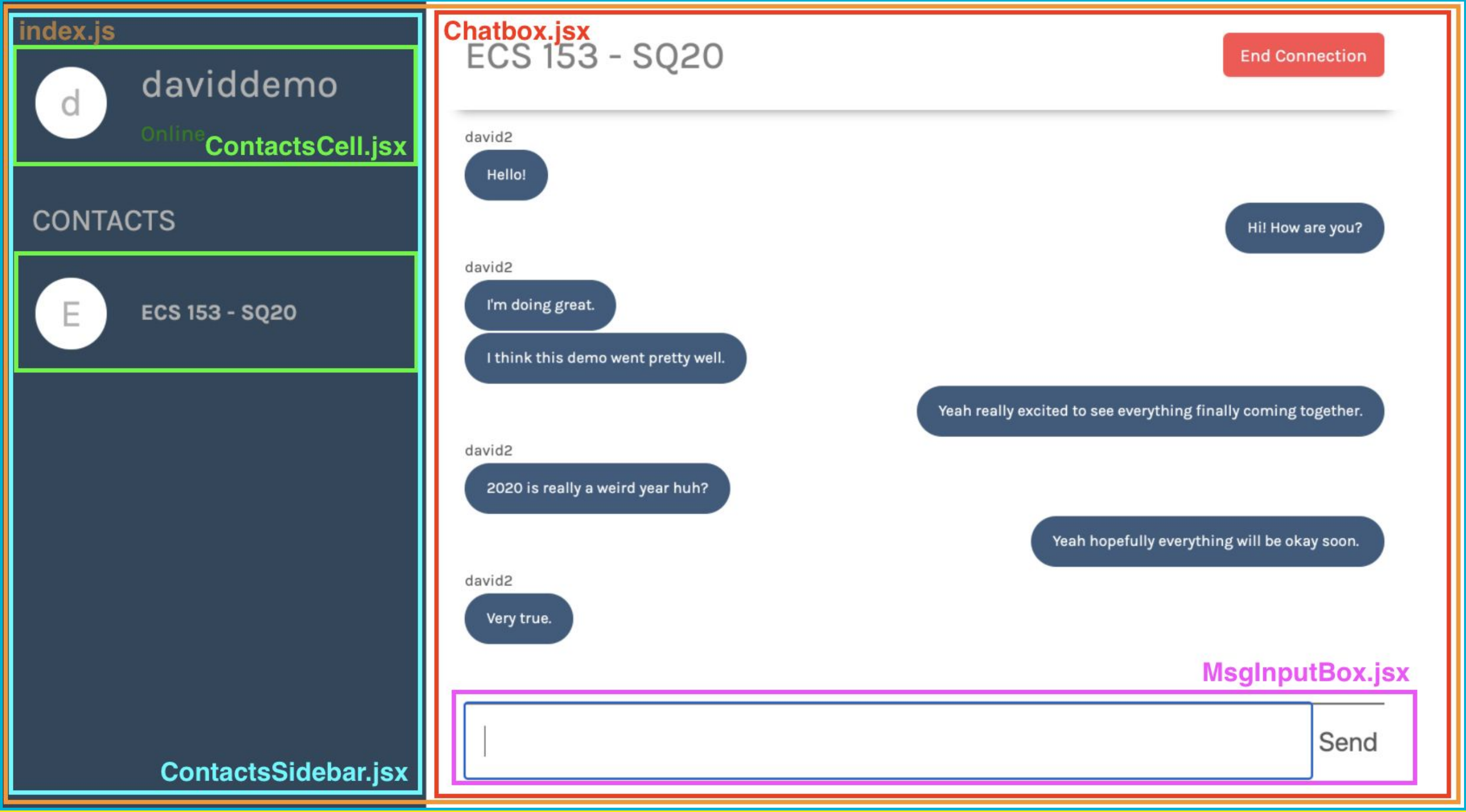


Username: chatsignal

Password: ••••••••••

Chatroom UI

- The chatroom is composed of several components for the purpose of UI reusability and state encapsulation:
    - Index.js - the entire chatroom itself that acts as the parent component.
    - ContactsSidebar.jsx - container for user’s personal info cell and contact list.
    - ContactsCell.jsx - represents an user information box
    - Chatbox.jsx - the entire right side of chatroom, containing chatroom’s header, end connection button, chatroom message display, and Message Input box
    - MsgInputBox.jsx - where user can enter text and send to other users via send button



david2

Hello!

Hi! How are you?

david2

I'm doing great.

I think this demo went pretty well.

Yeah really excited to see everything finally coming together.

david2

2020 is really a weird year huh?

Yeah hopefully everything will be okay soon.

david2

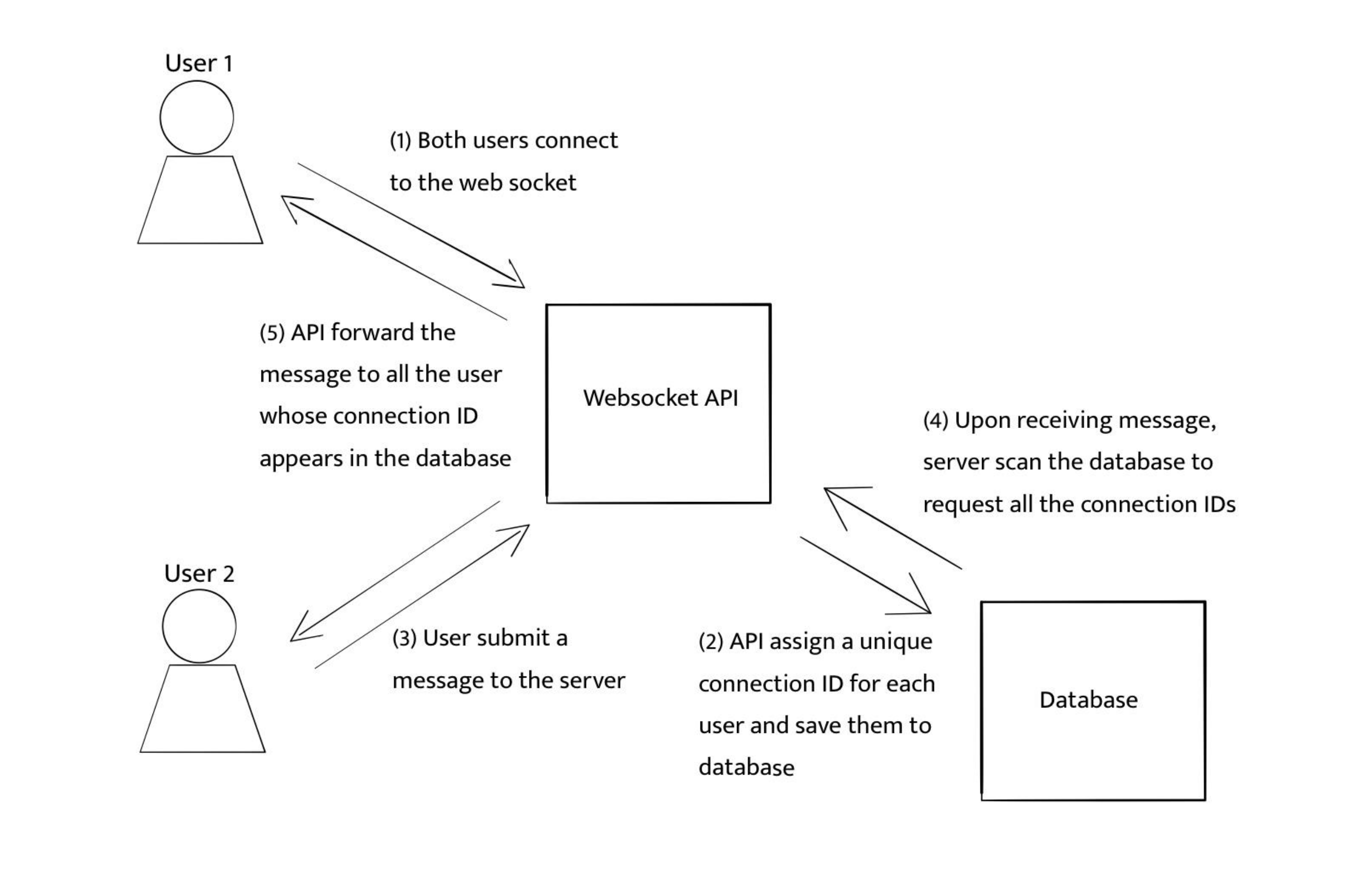
Very true.

Websocket

- Websocket is the backbone of our message app as it provides **real time** communication between clients.
- When a user connects to websocket, our API will assign them a unique connection ID and store the connection ID into a DB table.
- Users can then start sending messages to the websocket.
- API will forward the incoming messages to everyone whose connection ID appears in the same DB table.
- Upon disconnect, the API will scan the DB table and remove the disconnected user's connection ID.
- Encryption Key generation and sharing is also through websocket.

Websocket Performance

- A plain text message usually takes around less than a second to send to AWS and forward to everyone in the same room (When there are ~5 clients in the same room)
- Out front end code will fetch the whole message history array every 1 second from our backend.
- In total a message from send to display on UI usually takes ~1 to ~2 seconds.



(1) Both users connect to the web socket

(2) API assign a unique connection ID for each user and save them to database

(3) User submit a message to the server

(4) Upon receiving message, server scan the database to request all the connection IDs

(5) API forward the message to all the user whose connection ID appears in the database

Encryption

- AES Encryption with 256 bit key
- AES: Advanced Encryption Standard. Chosen as the standard for encrypting highly sensitive government information.
- Fast & Secure
- 256 bits means > $1.157920892 \times 10^{77}$ possible keys

AES Encryption & Decryption

AES_Encryption takes a message string and a BigInt key as arguments. The key is expanded using ExpandKey and the message string is split up into 16-character long groups. These groups are put into a 4x4 array, which is then encrypted by performing the following transformations:

- AddRoundKey

13 rounds of:

- SubBytes
- ShiftRows
- MixColumns

then:

- SubBytes
- ShiftRows
- AddRoundKey

AES_Decryption is meant to preform the inverse of all the Encryption transformations. It works the exact same way by splitting up the message, but each piece undergoes the following transformations instead:

- AddRoundKey

13 rounds of:

- invShiftRows
- invSubBytes
- invMixColumns

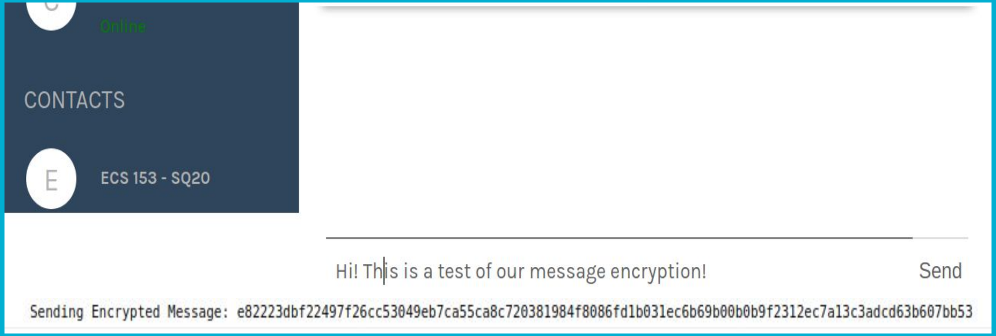
then:

- invShiftRows
- invSubBytes
- AddRoundKey

AES Test:

- The encryption and decryption algorithms works on 16 character blocks. It takes about 3.6 ms to encrypt each block of characters.

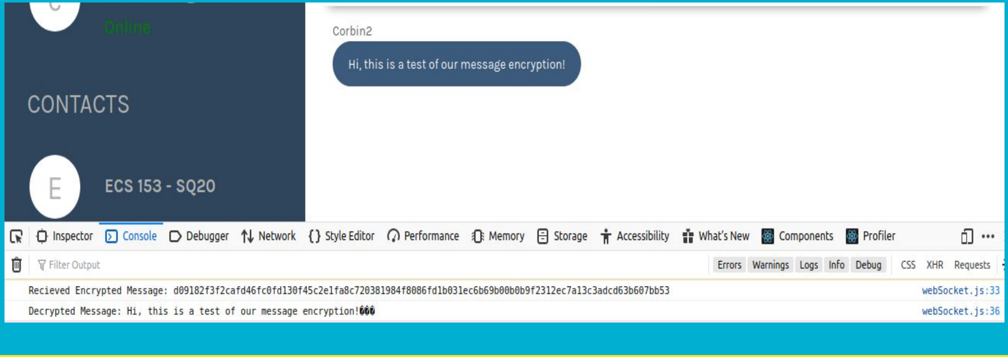
Sender:



Hi! This is a test of our message encryption!

```
Sending Encrypted Message: e82223dbf22497f26cc53049eb7ca55ca8c720381984f8086fd1b031ec6b69b00b0b9f2312ec7a13c3adcd63b607bb53
```

Receiver:



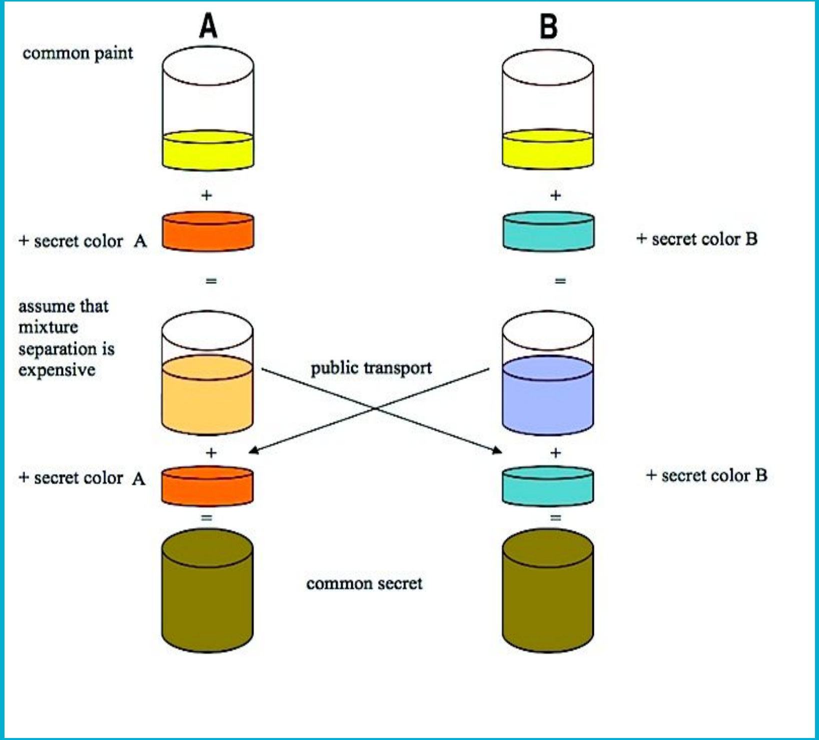
Corbin2

Hi, this is a test of our message encryption!

```
Recieved Encrypted Message: d09182f3f2cafd46fc0fd130f45c2e1fa8c720381984f8086fd1b031ec6b69b00b0b9f2312ec7a13c3adcd63b607bb53   webSocket.js:33
Decrypted Message: Hi, this is a test of our message encryption!���                                                            webSocket.js:36
```

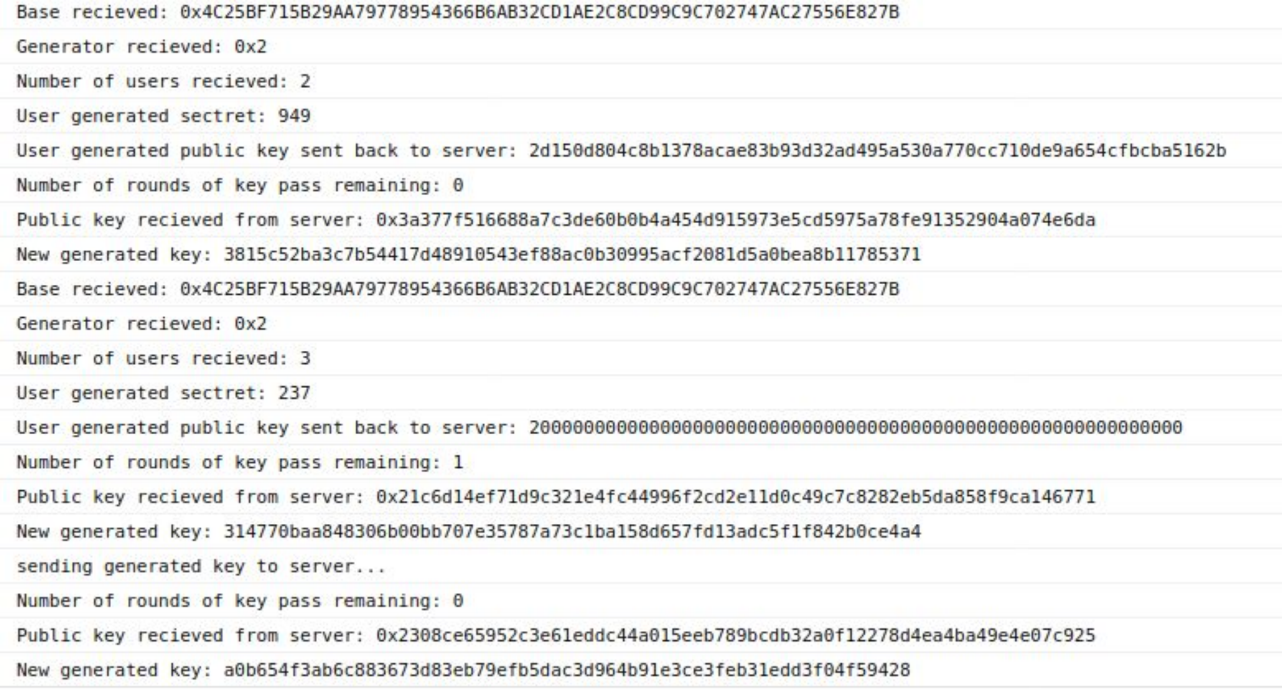
Key sharing

- Diffie-Hellman Key Exchange
- Each user holds an object that contain a secret, a generator (g), a base (p), a key and the number of rounds remaining before a private key is considered established.
- The secret of each user is a randomly generated number between 1 and 1000. This number is never shared with anyone. The generator and base are set by the server. The key is generated on each round of key passing.
- Keys are then passed in a circular fashion to another user
- Possible with any number of participants using a pass right policy



Key Exchange Test:

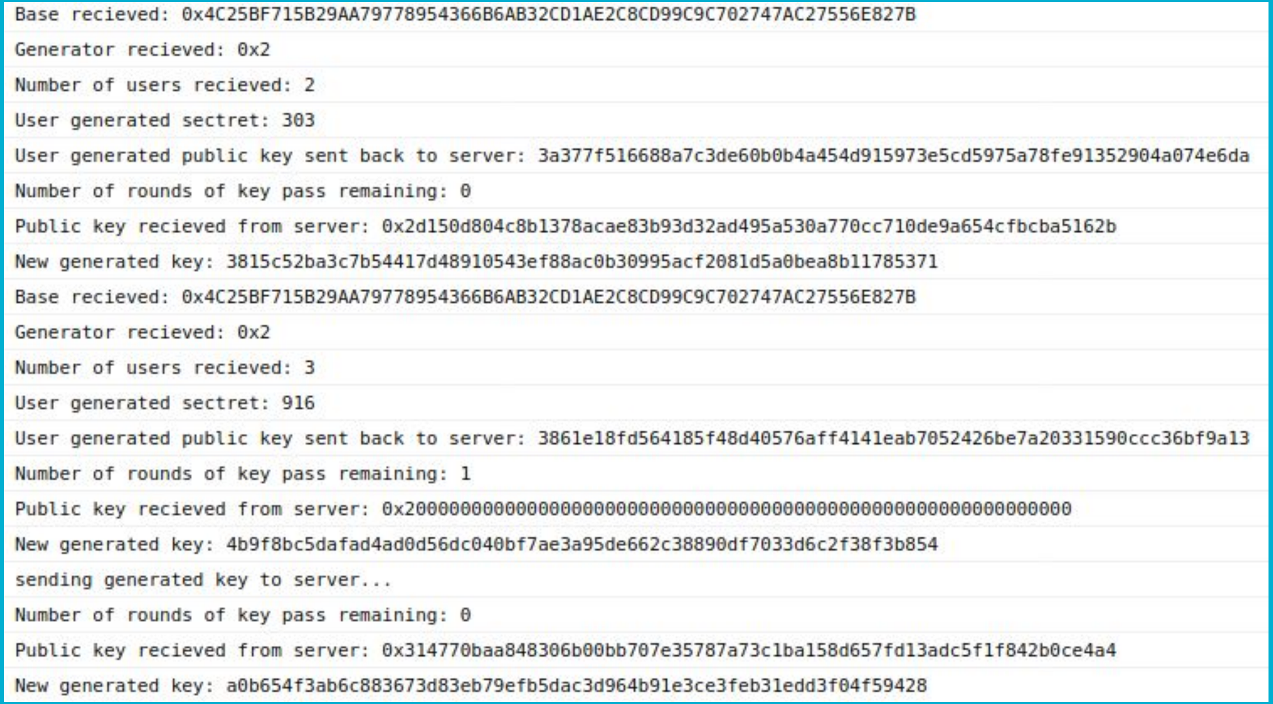
User 1:



```
Base recieved: 0x4C25BF715B29AA79778954366B6AB32CD1AE2C8CD99C9C702747AC27556E827B
Generator recieved: 0x2
Number of users recieved: 2
User generated sectret: 949
User generated public key sent back to server: 2d150d804c8b1378acae83b93d32ad495a530a770cc710de9a654cfbcba5162b
Number of rounds of key pass remaining: 0
Public key recieved from server: 0x3a377f516688a7c3de60b0b4a454d915973e5cd5975a78fe91352904a074e6da
New generated key: 3815c52ba3c7b54417d48910543ef88ac0b30995acf2081d5a0bea8b11785371
Base recieved: 0x4C25BF715B29AA79778954366B6AB32CD1AE2C8CD99C9C702747AC27556E827B
Generator recieved: 0x2
Number of users recieved: 3
User generated sectret: 237
User generated public key sent back to server: 200000000000000000000000000000000000000000000000000000000000
Number of rounds of key pass remaining: 1
Public key recieved from server: 0x21c6d14ef71d9c321e4fc44996f2cd2e11d0c49c7c8282eb5da858f9ca146771
New generated key: 314770baa848306b00bb707e35787a73c1ba158d657fd13adc5f1f842b0ce4a4
sending generated key to server...
Number of rounds of key pass remaining: 0
Public key recieved from server: 0x2308ce65952c3e61eddc44a015eeb789bcdb32a0f12278d4ea4ba49e4e07c925
New generated key: a0b654f3ab6c883673d83eb79efb5dac3d964b91e3ce3feb31edd3f04f59428
```

Key Exchange Test:

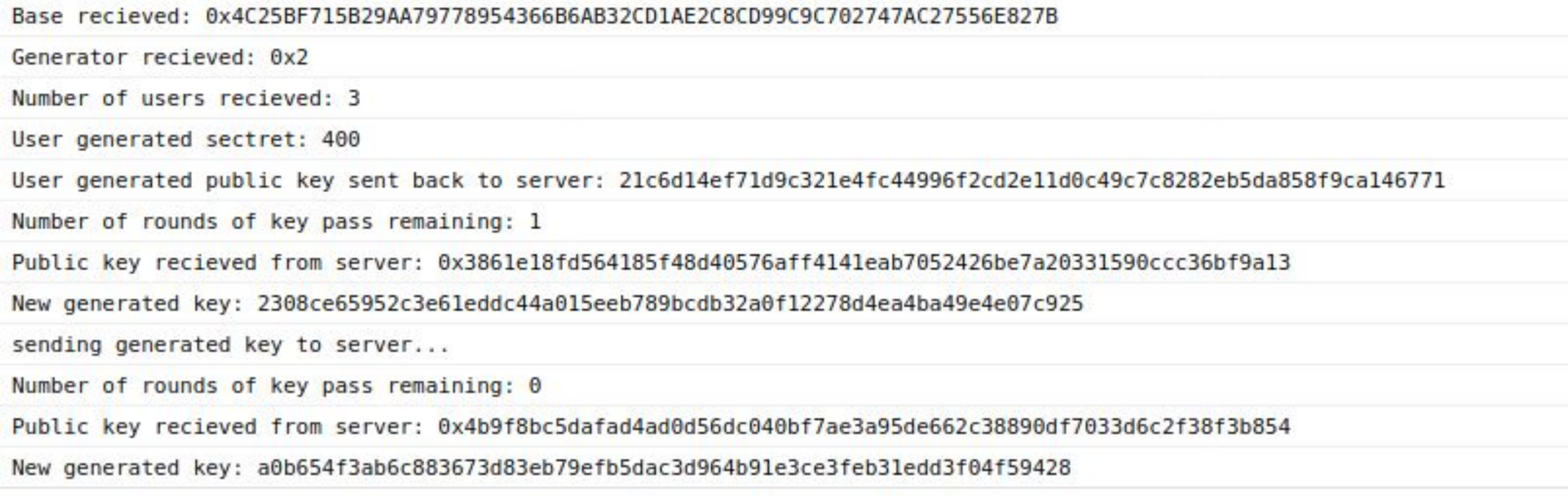
User 2:



```
Base recieved: 0x4C25BF715B29AA79778954366B6AB32CD1AE2C8CD99C9C702747AC27556E827B
Generator recieved: 0x2
Number of users recieved: 2
User generated sectret: 303
User generated public key sent back to server: 3a377f516688a7c3de60b0b4a454d915973e5cd5975a78fe91352904a074e6da
Number of rounds of key pass remaining: 0
Public key recieved from server: 0x2d150d804c8b1378acae83b93d32ad495a530a770cc710de9a654cfbcba5162b
New generated key: 3815c52ba3c7b54417d48910543ef88ac0b30995acf2081d5a0bea8b11785371
Base recieved: 0x4C25BF715B29AA79778954366B6AB32CD1AE2C8CD99C9C702747AC27556E827B
Generator recieved: 0x2
Number of users recieved: 3
User generated sectret: 916
User generated public key sent back to server: 3861e18fd564185f48d40576aff4141eab7052426be7a20331590ccc36bf9a13
Number of rounds of key pass remaining: 1
Public key recieved from server: 0x200000000000000000000000000000000000000000000000000000000000
New generated key: 4b9f8bc5dafad4ad0d56dc040bf7ae3a95de662c38890df7033d6c2f38f3b854
sending generated key to server...
Number of rounds of key pass remaining: 0
Public key recieved from server: 0x314770baa848306b00bb707e35787a73c1ba158d657fd13adc5f1f842b0ce4a4
New generated key: a0b654f3ab6c883673d83eb79efb5dac3d964b91e3ce3feb31edd3f04f59428
```

Key Exchange Test:

User 3: (joined after first two users)



```
Base recieved: 0x4C25BF715B29AA79778954366B6AB32CD1AE2C8CD99C9C702747AC27556E827B
Generator recieved: 0x2
Number of users recieved: 3
User generated sectret: 400
User generated public key sent back to server: 21c6d14ef71d9c321e4fc44996f2cd2e11d0c49c7c8282eb5da858f9ca146771
Number of rounds of key pass remaining: 1
Public key recieved from server: 0x3861e18fd564185f48d40576aff4141eab7052426be7a20331590ccc36bf9a13
New generated key: 2308ce65952c3e61eddc44a015eeb789bcdb32a0f12278d4ea4ba49e4e07c925
sending generated key to server...
Number of rounds of key pass remaining: 0
Public key recieved from server: 0x4b9f8bc5dafad4ad0d56dc040bf7ae3a95de662c38890df7033d6c2f38f3b854
New generated key: a0b654f3ab6c883673d83eb79efb5dac3d964b91e3ce3feb31edd3f04f59428
```

Final key agreed on by all users:

a0b654f3ab6c883673d83eb79efb5dac3d964b91e3ce
3feb31edd3f04f59428

Key Exchange Test:

- For 2 users, the average time to establish a secure key is: 280.9ms
- For 3 users: 726.4ms
- For 4 users: 1130.875ms
- For 5 users: 2145.1ms