Securing Social Media Applications

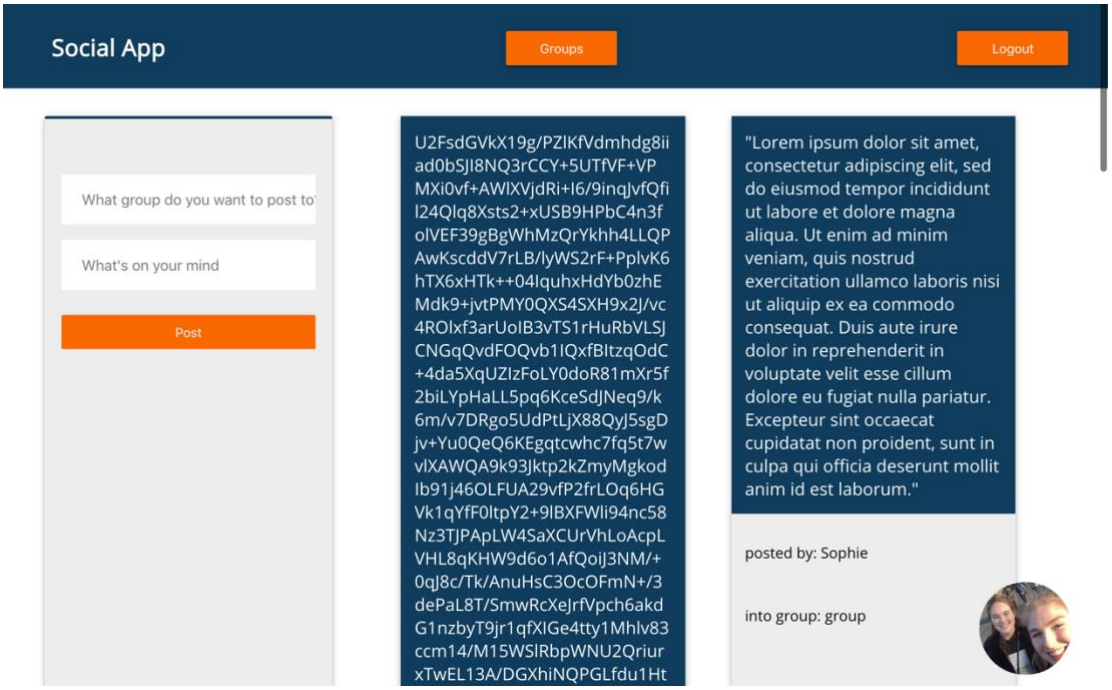
13 April 2020
CS34031
Advanced Telecommunications

Ruth Brennan
17329846

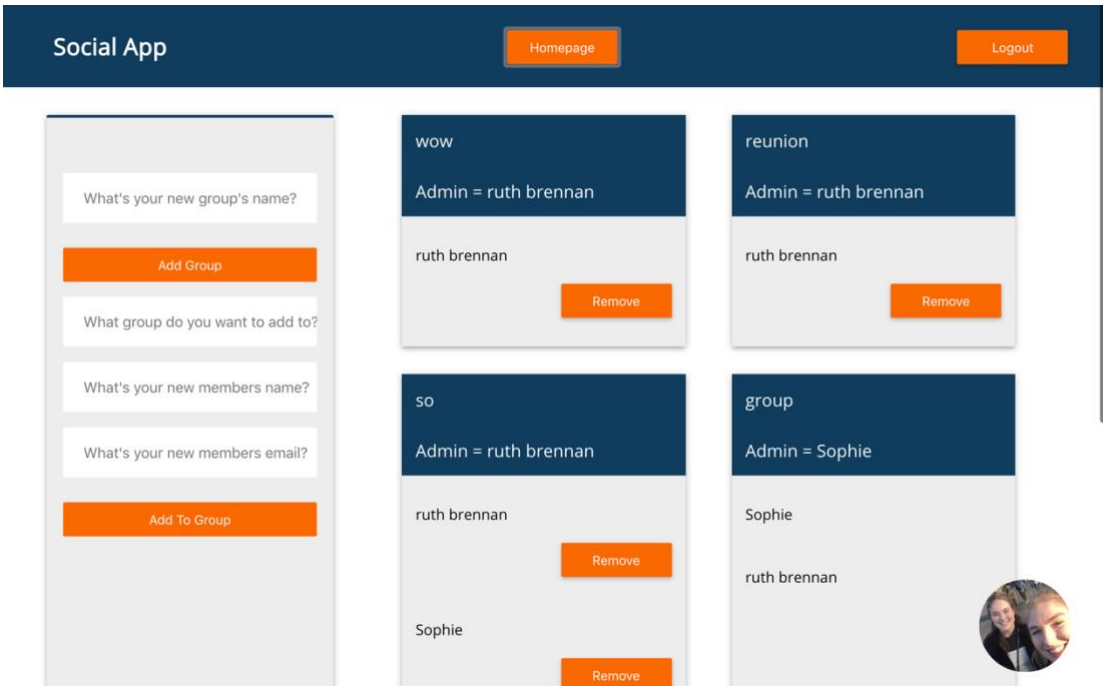
Approach

I decided to build my own social media application. It allows users to write posts to group of which they are a member. All posts can be seen by all users but only posts that were made to groups you are a member of are decrypted. The others are displayed as cipher text. You can remove (delete) any posts that you make. The application also allows users to make their own groups of which they will be an admin. You can add and remove members to groups that you created (ie. if you're the admin).

Here is the homepage where you can make, read and delete posts.



Here is the groups page where you can manage groups you are the admin of and look at the groups you are a part of.



Design

I built my web application using:

- React - A javascript library for building user interfaces - info
- Google firebase - I used firebase to help implement authentication and to act as a database - info
- cyrpto js – JavaScript library of cyrpto standards - info

Making Posts

The application allows the user to type in and make posts using the react form component. The user is asked what group they would like to post to. The constraints on this are that the group must exist and given that that they must be a member of the group.

The post is then encrypted using AES (implemented via the crypto js library) and the key using is generated using SHA1 to get a hash from the current date and a 'random' string.

```
//  *** Encryption ***
    // Here we generate a random key using today's date and a random string
    // The Cyrpto library allows us to user SHA1 to generate a hash that we use as the key
    var current_date = (new Date()).valueOf().toString();
    var random = Math.random().toString();
    var key = Crypto.createHash('sha1').update(current_date + random).digest('hex').toString();

    // Here we encrypt the post using AES which the Crypto Library provides
    // AES is plain text encryption
    var encryptedPost = CryptoJS.AES.encrypt(this.state.currentItem, key).toString();
```

Viewing Posts

All of the posts made on the application are displayed on the homepage. If the user is a member of the group to which a post was posted then that particular post is in clear text. If another post belongs to a group to which the current user is not a member then it will appear in cipher text.

The decryption is again preformed using crypto js and can only be decrypted by members of the group to which it was posted.

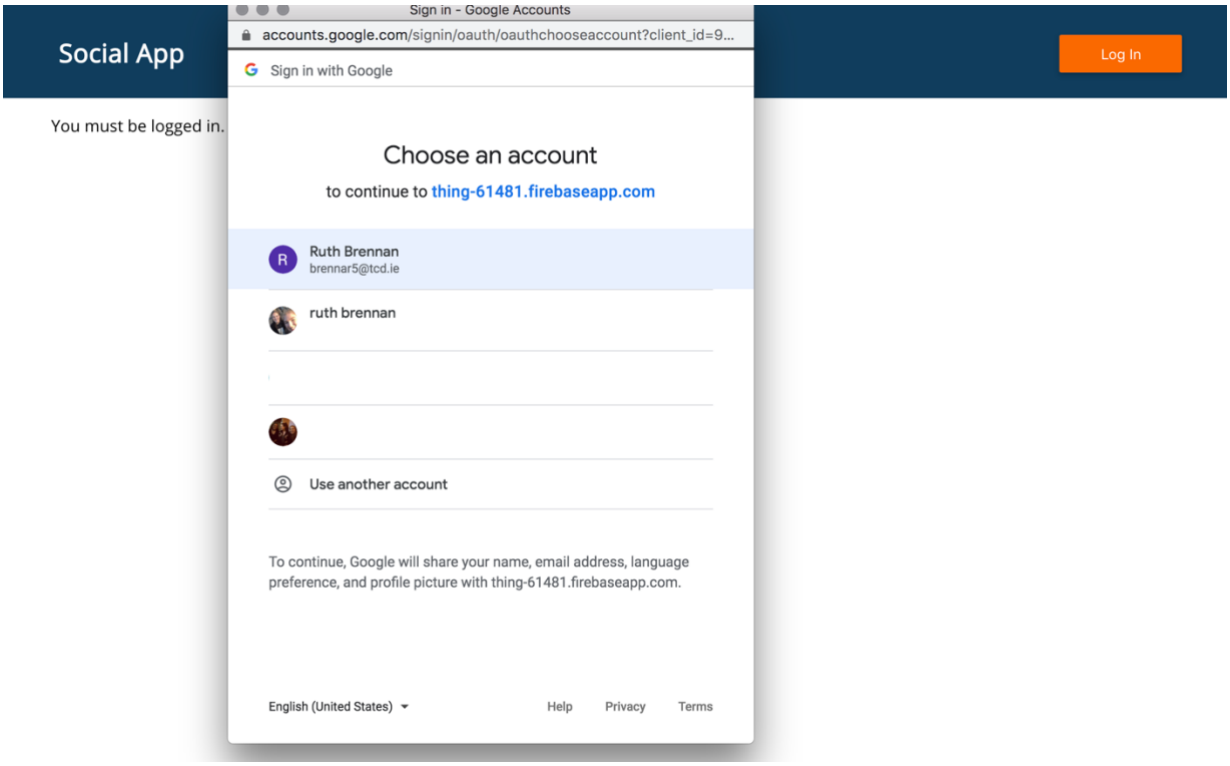
```
var originalText = CryptoJS.AES.decrypt(cipherText, key).toString(CryptoJS.enc.Utf8);
```

Groups

The application allows users to make their own groups and to act as the administrator for those groups. The administrator can add and remove members. They can also view groups that they are a member of.
The groups along with the posts and users are stored in the firebase database. They are updated using react forms and by calling the db using the reference to ‘items’ for posts etc.

Authentication

Using Firebase I was able to set the application up so that users must sign in with their google email account.



This is the login and logout functionality as it interacts with firebase.

```
//Method that uses auth from google firebase to log people in with their gmail
 login() {
  auth.signInWithPopup(provider)
    .then((result) => {
      const user = result.user;
      this.setState({
        user
      });
    });
}

 //Method that uses auth from google firebase to log people out
 logout() {
  auth.signOut()
```

```
    .then(() => {
      this.setState({
        user: null
      });
    });
}
```

Submission Parts

My submission is made up of

- This document (with appendix for the main code file – app.js)
- A demo video of how the application works - here
- A link to the GitHub Repository that hosts this project - here

Appendix – App.js

```
import React, { Component } from 'react';
import './App.css';

//  Used to store posts, user info etc and to authenticate user login with google accounts
//  https://firebase.google.com/
import firebase, { auth, provider } from './firebase.js';

//  Library used to encrypt posts
//  https://www.npmjs.com/package/crypto-js
var Crypto = require("crypto");
var CryptoJS = require("crypto-js");


class App extends Component {

  constructor() {

    super();

    //state holds all global variables whose change should call for a refresh
    this.state = {
      currentItem: '',
      username: '',
      user: null,
```

```
    items:[],
    homepage: true,
    groupName: '',
    memberToGroup: '',
    memberName: '',
    groups: [],
    postToGroup: '',
    memberEmail:'',
  }

  //binding all the methods to this
  this.handleChange = this.handleChange.bind(this);
  this.handleSubmit = this.handleSubmit.bind(this);
  this.login = this.login.bind(this);
  this.logout = this.logout.bind(this);
  this.handleAddGroup = this.handleAddGroup.bind(this);
  this.handleAddToGroup = this.handleAddToGroup.bind(this);
  this.switchPage = this.switchPage.bind(this);
  this.text = this.text.bind(this);
  this.decrypt = this.decrypt.bind(this);
}

//Method that uses auth from google firebase to log people in with their gmail
login() {
  auth.signInWithPopup(provider)
    .then((result) => {
      const user = result.user;
      this.setState({
        user
      });
    });
}

//Method that uses auth from google firebase to log people out
logout() {
  auth.signOut()
    .then(() => {
      this.setState({
        user: null
      });
    });
}
```

```
//Method to handle posting
// *** Encryption is done here ***
handleSubmit(e) {

    //we need to prevent the default behavior of the form,
    //which if we don't will cause the page to refresh when you hit the submit button
    e.preventDefault();

    //We need to carve out a space in our Firebase database where we'd like to store all of the posts.
    //We do this by calling the ref method and passing in the destination we'd like them to be stored (items).
    const itemsRef = firebase.database().ref('items');

    //  *** Encryption ***
    // Here we generate a random key using today's date and a random string
    // The Cyrpto library allows us to user SHA1 to generate a hash that we use as the key
    var current_date = (new Date()).valueOf().toString();
    var random = Math.random().toString();
    var key = Crypto.createHash('sha1').update(current_date + random).digest('hex').toString();

    // Here we encrypt the post using AES which the Crypto Library provides
    // AES is plain text encryption
    var encryptedPost = CryptoJS.AES.encrypt(this.state.currentItem, key).toString();
    console.log('encrypt: ');
    console.log(encryptedPost);

    //  Make item object to represent the post
    //  The CipherText is stored, as is the group to which it is posted, the user who posted it and the key to decrypt it
    const item = {
      title: encryptedPost,
      group: this.state.postToGroup,
      user : this.state.user.displayName,
      key: key,
    }

    //  Here we add the new item to the items on the firebase as well and reset the state values used to create this new item
    //  The local items will be updated in components did mount
    itemsRef.push(item);
    this.setState({
      currentItem: '',
      postToGroup: '',
    });
}
```

```
 //  *** Decryption ***
//  This is the function where decrytion is preformed
decrypt(cipherText, key) {
  var originalText = CryptoJS.AES.decrypt(cipherText, key).toString(CryptoJS.enc.Utf8);
  return originalText;
}

//  This function checks if the current user is able to view the post as cipher text or not
text(item) {

  //Getting Groups from db
  let decrypt = false;  // This boolean tells us to decrypt or not
  const groupRef = firebase.database().ref('groups');
  groupRef.on('value', (snapshot) => {
    let tempGroups = snapshot.val();
    for (let group in tempGroups) {
      if (tempGroups[group].name == item.group) {
        if (tempGroups[group].members.includes(this.state.user.displayName)) {
          decrypt=true;
        }
      }
    }
  });

  if (decrypt) {
    var hold = this.decrypt(item.title,item.key);
    console.log('hold ' + hold);
    return (hold);
  } else {
    console.log('item.title ' + item.title);
    return(item.title);
  }
}

//  This function handles adding a new group
//  This is very similar to handle submit as this is also called from a form
//  The differenece here is that we add a group as opposed to a post
handleAddGroup(e) {

  e.preventDefault();

  const groupsRef = firebase.database().ref('groups');
```

```
  var memberList = [this.state.user.displayName];

  const group = {
    name: this.state.groupName,
    members: memberList,
    admin: this.state.user.displayName
  }

  groupsRef.push(group);
  this.setState({
    groupName: '',
  });

}

//  This function handles adding a new member to a group
//  This is very similar to handle submit as this is also called from a form
//  The differenece here is that we add a member to a group
handleAddToGroup(e) {

  e.preventDefault();

  let newGroups = '';
  //  Getting Groups from db
  const groupRef = firebase.database().ref('groups');
  groupRef.on('value', (snapshot) => {
    newGroups = snapshot.val();
    for (let group in newGroups) {
      console.log(newGroups[group].name);
      //  We find the group that we want to add to
      if (newGroups[group].name == this.state.memberToGroup) {
        //  We add the member to our local temporary group
        newGroups[group].members.push(this.state.memberName);
      }
    }
  });

  //  Here we set as opposed to post
  groupRef.set(newGroups);

  this.setState({
    memberToGroup: '',
    memberName: '',
```

```
    memberEmail: '',
  });

}

// This is a simple multipurpose function that sets a value in the state based on where it was called from
handleChange(e) {
    this.setState({
      [e.target.name]: e.target.value
    });
}


//  We'll attach this event listener inside of our componentDidMount
componentDidMount() {

  //  Login stuff
  auth.onAuthStateChanged((user) => {
    if (user) {
      this.setState({ user });
    }
  });
  //  Getting items (posts) from db
  const itemsRef = firebase.database().ref('items');
  itemsRef.on('value', (snapshot) => {
    let items = snapshot.val();
    let newState = [];
    for (let item in items) {
      newState.push({
        id: item,
        title: items[item].title,
        user: items[item].user,
        group: items[item].group,
        key: items[item].key
      });
    }
    this.setState({
      items: newState
    });
  });

  //  Getting Groups from db
  const groupRef = firebase.database().ref('groups');
```

```
  groupRef.on('value', (snapshot) => {
    let groups = snapshot.val();
    let newState = [];
    for (let group in groups) {
      newState.push({
        id: group,
        name: groups[group].name,
        members: groups[group].members,
        admin: groups[group].admin
      });
    }
    this.setState({
      groups: newState
    });
  });

}

//  Function to remove a post
removeItem(itemId) {
  const itemRef = firebase.database().ref(`/items/${itemId}`);
  itemRef.remove();
}


removeGroup() {
  //TODO
}

//   Remove a member from a group
//   This can only be done by a group admin
handleRemoveMember(memberName, groupName) {

  let newGroups = '';
  //  Getting Groups from db
  const groupRef = firebase.database().ref('groups');
  groupRef.on('value', (snapshot) => {
    newGroups = snapshot.val();
    for (let group in newGroups) {
      console.log(newGroups[group].name);
      if (newGroups[group].name == groupName) {
        if (newGroups[group].admin != memberName) {
          var index = newGroups[group].members.indexOf(memberName);
```

```
          newGroups[group].members.splice(index,1);
        }
      }
    }
  });

  groupRef.set(newGroups);
  this.setState({
    memberToGroup: '',
    memberName: '',
  });

}

//  This just changes our global variable so that we can switch between
//  the homepage (for posting) and the groups page (to manage groups)
switchPage(){
  let temp = !this.state.homepage;
  this.setState({
    homepage: temp,
  });
}

//  This is the render that actually puts things on the screen
render() {
  return (
    //  This is the header part
    //  So our banner, 'logo', logout and switch page buttons
    <div className='app'>
      <header>
        <div className="wrapper">
          <h1>Social App</h1>
          {this.state.homepage ?
            <button onClick={this.switchPage}>Groups</button>
            :
            <button onClick={this.switchPage}>Homepage</button>
          }
          {this.state.user ?
            <button onClick={this.logout}>Logout</button>
            :
            <button onClick={this.login}>Log In</button>
          }
        </div>
```

```
      </header>

      { /* This then makes up the body of the page */ }

      { /* Here we make sure that the user is logged in */ }
      {this.state.user ?
        <div>
          {/* Then we display the contents based on what page we are on */ }
          { this.state.homepage ?
            // Here we are on the homepage
            <div>
              {/* Displays the users picture based on their gmail profile pic */ }
              <div className='user-profile'>
                <img src={this.state.user.photoURL} />
              </div>
              {/* This takes care of writing posts */ }
              <div className='container'>
                <section className='add-item'>
                  <form onSubmit={this.handleSubmit}>
                    <input type="text" name="postToGroup" placeholder="What group do you want to post to?"
onChange={this.handleChange} value={this.state.postToGroup} />
                    <input type="text" name="currentItem" placeholder="What's on your mind" onChange={this.handleChange}
value={this.state.currentItem} />
                    <button>Post</button>
                  </form>
                </section>
                {/* This takes care of displaying posts */ }
                <section className='display-item'>
                  <div className="wrapper">
                    <ul>
                      {this.state.items.map((item) => {
                        return (
                          <li key={item.id}>
                            <h3>{ this.text(item) }</h3>
                            <p>posted by: {item.user}</p>
                            <p> into group: {item.group}
                              {item.user === this.state.user.displayName || item.user === this.state.user.email ?
                                <button onClick={() => this.removeItem(item.id)}>Remove Post</button> : null}
                            </p>
                          </li>
                        )
                      })}
                    </ul>
```

```
            </div>
          </section>
        </div>
      </div>
      :
      //We are on the group page
      <div>
        <div className='user-profile'>
          <img src={this.state.user.photoURL} />
        </div>
        {/* This takes care of adding groups where you will be the admin */ }
        <div className='container'>
          <section className='add-item'>
            <form onSubmit={this.handleAddGroup}>
              <input type="text" name="groupName" placeholder="What's your new group's name?" onChange={this.handleChange}
value={this.state.groupName} />
              <button>Add Group</button>
            </form>
            <section>
              <p></p>
            </section>
            <section>
              <p></p>
            </section>
            {/* This takes care of adding people to groups of which you are the admin */ }
            <form onSubmit={this.handleAddToGroup}>
              <input type="text" name="memberToGroup" placeholder="What group do you want to add to?"
onChange={this.handleChange} value={this.state.memberToGroup} />
              <input type="text" name="memberName" placeholder="What's your new members name?"
onChange={this.handleChange} value={this.state.memberName} />
              <input type="text" name="memberEmail" placeholder="What's your new members email?"
onChange={this.handleChange} value={this.state.memberEmail} />
              <button>Add To Group</button>
            </form>
          </section>
          <section className='display-item'>
            <div className="wrapper">
              {/* Displays the groups that you are a part of */ }
              <ul>
                {this.state.groups.map((item) => {
                  if (item.members.includes(this.state.user.displayName) ) {
                    return (
                      <li key={item.id}>
```

```
                        <h3>{item.name}</h3>
                        <h3>Admin = {item.admin}</h3>
                        {
                          item.members.map((member) => {
                            if (this.state.user.displayName == item.admin) {
                              return(
                                <div>
                                  {/* If you are an admin then you have the option to remove people from that group */ }
                                  <p>{member}
                                      <button onClick={() => this.handleRemoveMember(member, item.name)}>Remove</button>
                                  </p>
                                </div>
                              );
                            } else {
                              return(
                                <div>
                                  <p>{member}</p>
                                </div>
                              );
                            }
                            })}
                        </li>
                    ))))}
                  </ul>
                </div>
              </section>
            </div>
          </div>
        }
        </div>
        :
        <div className='wrapper'>
          <p>You must be logged in.</p>
        </div>
      }

    </div>
  );
 }
}
export default App;
```